Observabilidade em DevSecOps:

Visualização de Dados de Segurança de Aplicações Web

Mestrado em Ciência de Dados

Relatório de Estágio

Alexandre Emanuel Carriço Penela

Leiria, março de 2025

Observabilidade em DevSecOps: Visualização de Dados de Segurança de Aplicações Web

Mestrado em Ciência de Dados

Relatório de Estágio

Alexandre Emanuel Carriço Penela

Estágio realizado sob a orientação da Professora Doutora Maria Beatriz Guerra da Piedade e da Professora Doutora Rosa Isabel Alves Cordeiro Matias e sob supervisão de Bernardo Sequeira.

Leiria, março de 2025

Originalidade e Direitos de Autor

O presente relatório de estágio é original, elaborado unicamente para este fim, tendo sido devidamente citados todos os autores cujos estudos e publicações contribuíram para o elaborar.

Reproduções parciais deste documento serão autorizadas na condição de que seja mencionado o Autor e feita referência ao ciclo de estudos no âmbito do qual o mesmo foi realizado, a saber, Mestrado em Ciência de Dados, no ano letivo 2023/2024 da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria, Portugal, e, bem assim, à data das provas públicas que visaram a avaliação destes trabalhos.

Agradecimentos

Com este relatório mais um ciclo se encerra. Levo não apenas todo o conhecimento adquirido, mas também um conjunto de experiências que nunca esquecerei. Este relatório representa o culminar de muito esforço, empenho, suor e algumas lágrimas naquele que foi um percurso desafiante. Através dele foi possível retirar ensinamentos, não apenas para aplicar no contexto profissional, mas também para serem usados na vertente pessoal.

Quero agradecer assim, a todas as pessoas e organizações, que, de uma forma ou de outra, contribuíram para o sucesso deste trabalho.

Começo por agradecer às minhas orientadoras, a Professora Doutora Maria Beatriz Piedade e a Professora Doutora Rosa Matias pelo acompanhamento, paciência, compreensão e partilha de conhecimento ao longo de todo o percurso. O meu "muito obrigado".

Quero expressar um enorme agradecimento à *VOID Software, S.A*. pela oportunidade que me deu de poder realizar este estágio, assim como todo o suporte e flexibilidade fornecido para que todo o trabalho de pudesse ser concluído.

Dentro da *VOID Software*, quero agradecer ao meu supervisor Bernardo Sequeira, ao Eduardo Piza e ao Fábio Henriques, por todo o auxílio fornecido e por me guiarem durante a execução de todas as tarefas de estágio. Quero ainda dar um agradecimento especial à Catarina Reis, que considero como uma mentora, cujo apoio, disponibilidade e *feedback* contínuos foram essenciais tanto para execução das atividades do estágio, como para a escrita deste documento.

Aos colegas de mestrado, em especial à Susana, à Beatriz, à Diana e ao João Peixoto, por todos os momentos e toda a convivência passada, dentro e fora de aulas. Foi (e continua a ser) uma sorte e um gosto ter-vos conhecido. Obrigada por tudo.

Por fim, um agradecimento muito especial aos meus pais, que foram a minha base de apoio emocional, nos momentos mais desafiantes deste percurso. Obrigado pelo incentivo, por toda a paciência, por toda a compreensão, por me puxarem para a razão quando o lado emocional prevalecia, por me ajudarem a manter o foco, quando este teimava a desvanecer. Acima de tudo, obrigado por acreditarem sempre em mim.

Muito, muito obrigado!

Resumo

O crescimento excecional do número de atividades de cibercrime que exploram as vulnerabilidades existentes em aplicações Web reforçou a necessidade das organizações em fortalecerem as suas políticas de segurança em vigor durante o ciclo de vida de desenvolvimento de software – SDLC. Para suprir esta necessidade surgiu o conceito de DevSecOps, que implementa práticas e testes de segurança, realizados por *scanners* automáticos, durante todo o SDLC. Estes scanners possuem meios para reportar as vulnerabilidades identificadas, mas geralmente apresentam funcionalidades limitadas para representar essa informação de forma visual.

Este documento apresenta a descrição de uma solução desenvolvida no âmbito de um estágio curricular realizado em ambiente empresarial. A solução permite a extração, integração e visualização de dados relevantes e constantes nos relatórios de diversos tipos ferramentas de segurança. Os dados extraídos e integrados são apresentados graficamente, por meio de um *dashboard* interativo e customizado, explorando os conceitos associados à visualização de dados. São descritas todas etapas de desenvolvimento desta solução, desde o processo de extração de dados até ao desenvolvimento do *dashboard*.

A análise comparativa efetuada demonstra que a solução se destaca das soluções semelhantes, devido à sua capacidade de integração e apresentação de dados de múltiplos tipos de ferramentas de segurança num *dashboard* unificado, contrariamente às restantes, que se focam em tipos específicos de ferramentas de seguranças. Esta abordagem proporciona uma visão geral do estado da segurança das aplicações e uma análise detalhada das vulnerabilidades existentes.

O contributo principal desta solução reside na melhoria do processo de monitorização e avaliação do estado de segurança das aplicações desenvolvidas, fornecendo *insights* valiosos para a organização. Esses *insights* auxiliam na tomada de decisões sobre a implementação de medidas e protocolos de segurança no SDLC. Este contributo foi corroborado pela realização de um inquérito SUS, que confirma a validade da solução desenvolvida.

Palavras-chave: Aplicações Web, *Dashboard*, Ferramentas de Segurança, Segurança de Aplicações, Visualização de Dados, Vulnerabilidades

Abstract

The exceptional rise in cybercriminal activities exploiting web application vulnerabilities has intensified the need for organizations to strengthen their security policies throughout the Software Development Life Cycle (SDLC). The concept of DevSecOps emerged to address this issue, integrating security practices and automated security testing using scanners throughout the entire SDLC. These scanners have mechanisms for reporting identified vulnerabilities but have limited capabilities for visually representing this information in general.

This document presents the description of a solution developed as part of a curricular internship, conducted in a corporate environment. The solution enables the extraction, integration and visualization of relevant data contained in reports from several types of security tools. The extracted and integrated data are presented graphically through an interactive and customizable dashboard, leveraging concepts associated with data visualization. All development stages of this solution are described, from the data extraction process to the dashboard implementation.

The comparative analysis conducted demonstrates that this solution stands out from similar alternatives due to its ability to integrate and present data from multiple types of security tools in a unified dashboard, unlike existing solutions that focus on specific types of security tools. This approach provides an overall view of the security status of applications, offering a detailed analysis of existing vulnerabilities.

The main contribution of this solution lies in improving the process of monitoring and assessing the security status of developed applications, providing valuable insights for the organization. These insights support decision-making regarding the implementation of security measures and protocols throughout the application development cycle. This contribution was validated through a SUS survey, which confirms the effectiveness of the developed solution.

Keywords: Web Applications, Dashboard, Security Tools, Application Security, Data Visualization, Vulnerabilities

Índice

Lista de Figuras

Lista de Tabelas

Lista de Listagens

Lista de Siglas e Acrónimos

BDD	Behaviour Driven Development
BI	Business Intelligence
CD	Continuous Delivery
CI	Continuous Integration
CSV	Comma-Separated Values
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
DAST	Dynamic Application Security Testing
ESTG	Escola Superior de Tecnologia e Gestão
ETL	Extract, Transform, Load
GUI	Graphical User Interface
HCI	Human - Computer Interaction
IA	Inteligência Artificial
IoT	Internet of Things
JSON	JavaScript Object Notation
KPI	Key Performance Indicators
LCP	Low Code Platform
OWASP	Open Web Application Security Project
PRISMA	Preferred Reporting Items for Systematic Reviews and Meta-Analysis
RGPD	Regulamento Geral de Proteção de Dados
RSL	Revisão Sistemática de Literatura
SAST	Static Application Security Testing
SBOM	Software Bill of Materials
SCA	Software Component Analysis
SDLC	Software Development Life Cycle
SGI	Sistema de Gestão Integrado
SQL	Structured Query Language
SSRF	Server-Side Request Forgery
SUS	System Usability Scale
UE	União Europeia

WASC	Web Application Security Consortium
WSAP	Web Security Automation Project

1. Introdução

O presente relatório foi elaborado com o objetivo de descrever as etapas do trabalho desenvolvido no âmbito da unidade curricular de estágio do Mestrado em Ciência de Dados, da Escola Superior de Tecnologia e Gestão (ESTG) [1] do Politécnico de Leiria [2], que foi realizado na empresa *VOID SOFTWARE, S.A.* [3].

1.1. Motivação e objetivos

Os dados são hoje reconhecidos pela maioria das grandes empresas e organizações como valiosos, devido à informação e conhecimento que pode ser obtido através destes, prestando auxílio às organizações na tomada de decisão e consequentemente conferindo-lhes vantagens competitivas. A visualização de dados é um dos métodos de análise da informação que através da representação e análise visual, possibilita a compreensão de informação complexa, a identificação de padrões, tendências e anomalias de maneira intuitiva. Essa abordagem complementa outras técnicas analíticas, ajudando a transformar os dados em *insights* valiosos.

Num mundo cada vez mais digitalizado, onde utilizadores e organizações estão mais dependentes de aplicações, estas têm-se tornado um alvo crescente de atividades enquadradas no cibercrime. De acordo com um relatório de investigação publicado pela Verizon em 2023 [4], as aplicações Web foram o principal vetor de ação utilizado por criminosos em atividades de cibercrime.

Implementar medidas de segurança no processo de desenvolvimento das aplicações torna-se assim, uma necessidade imperativa, mitigando desde cedo, a existência de vulnerabilidades e protegendo informação sensível. Esta necessidade levou ao aparecimento do conceito de DevSecOps, que une Desenvolvimento (Dev), Segurança (Sec) e Operações (Ops). O conceito integra um conjunto de práticas e medidas de segurança durante todo o ciclo de vida do desenvolvimento de aplicações [5, 6].

Uma das práticas comuns de DevSecOps é a utilização de *scanners* que realizam testes de segurança automáticos, permitindo deste modo, identificar vulnerabilidades nas fases iniciais de desenvolvimento, onde a correção de falhas é mais rápida e a um custo mais baixo para a empresa. Embora os *scanners* de segurança possuam mecanismos que reportam as vulnerabilidades identificadas e as suas características, eles frequentemente apresentam limitações na representação visual dessa informação, seja pela ausência de elementos gráficos nos relatórios gerados ou por oferecerem somente visualizações simples, com recursos limitados em termos de cores e configurações [7]. Adicionalmente, a maioria destas ferramentas não permite a integração de dados entre si, nem incorpora uma dimensão temporal na análise dos dados apresentados.

Uma análise eficaz e integrada dos dados dos relatórios gerados pelos diferentes tipos de *scanners* de segurança contribui para uma melhor monitorização e avaliação do estado de segurança das aplicações, fornecendo *insights* valiosos tanto para a equipa de segurança como para a gestão de topo, auxiliando na tomada de decisão relativa a ajustes e/ou implementação de medidas e protocolos de segurança no ciclo de desenvolvimento de aplicações.

O objetivo principal do presente estágio assenta, portanto, no desenvolvimento de uma solução que alia as capacidades e vantagens da visualização de dados ao paradigma de DevSecOps, por meio da integração de dados relevantes extraídos dos relatórios gerados pelas diversas ferramentas de segurança adotadas na *VOID SOFTWARE S.A.*, organização de acolhimento do estágio, e da sua representação visual, por meio de um *dashboard* interativo e customizado. Desta forma, a abordagem permite obter uma visão holística do estado da segurança das aplicações, desenvolvidas pela empresa, proporcionando uma análise detalhada das vulnerabilidades existentes. Adicionalmente, a incorporação de um mecanismo de inteligência temporal possibilita uma análise evolutiva do estado global de segurança das várias aplicações ao longo do tempo. Para mais detalhes, a proposta inicial de estágio pode ser consultada no Anexo A.

1.2. Caracterização da Entidade de Acolhimento

Desde o início da sua atividade em 2006, a *VOID SOFTWARE S.A*. dedica-se ao desenvolvimento de software, criando soluções adaptadas às necessidades específicas de cada desafio, recorrendo para isso, a uma vasta gama de tecnologias de programação. A empresa possui uma abordagem ágil centrada na criação de produtos personalizados, de acordo com as necessidades individuais do cliente, privilegiando a experiência emocional do utilizador [8].

Com clientes localizados em várias regiões do mundo, a *VOID SOFTWARE* tem vindo a adicionar valor nas mais diversas áreas, como o marketing, o sector contabilístico e financeiro, passando pelo desenvolvimento de sistemas de gestão de ativos digitais, até à construção e manutenção de plataformas de gestão integrada de entidades públicas e privadas de grande dimensão.

A *VOID SOFTWARE* tem apostado no desenvolvimento de produtos inovadores (próprios ou em parceria) em áreas como a Inteligência Artificial (IA), *Machine Learning* e *Blockchain*, contando com várias participações em *startups* e consórcios de base tecnológica. Implementa serviços de *cloud*, *Software as a Service* (SaaS), integração de *hardware*, aplicações *web* e de dispositivos móveis [8].

1.2.1. Organograma

A *VOID SOFTWARE S.A.* atualmente emprega mais de 90 colaboradores, de modo a fazer face às necessidades individuais de cada tarefa, permitindo a mobilização de grupos interdisciplinares, composta por vários departamentos que estão apresentados de acordo com as suas relações e hierarquia na Figura 1.

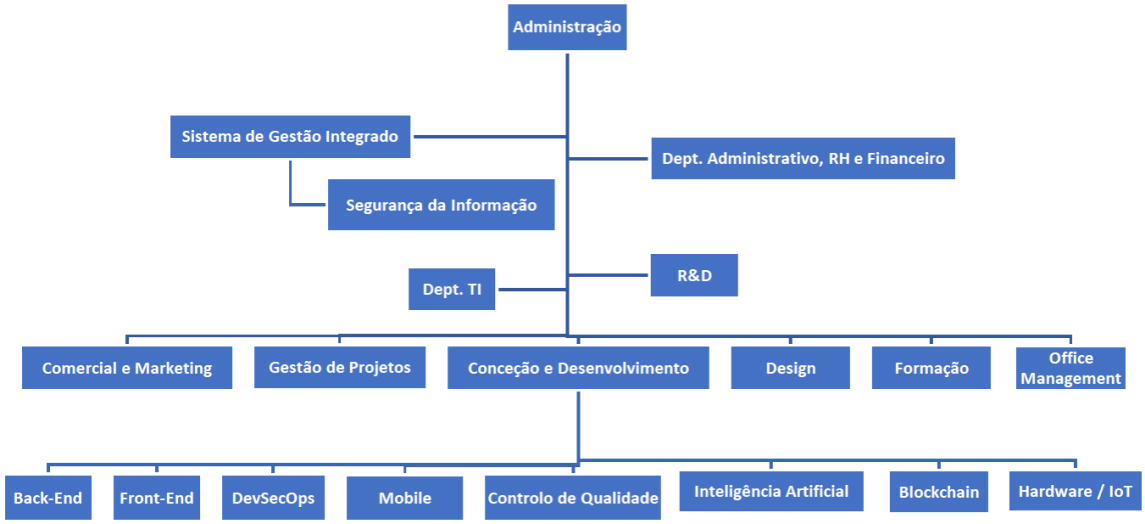


Figura 1 – Organograma da *VOID SOFTWARE*

1.2.2. Principais marcos históricos

Desde 2006 até à atualidade a *VOID SOFTWARE* possui como marcos históricos principais (informação proveniente de documentação interna da empresa):

- 2006: Fundação da VOIDsteam, Lda.
- 2013: Primeiros projetos a nível internacional dentro da EU.
- 2016: Primeiros projetos a nível internacional fora da UE; Participação na 1ª edição do *WebSummit.*
- 2018: Transformação da empresa em Sociedade Anónima (*VOID SOFTWARE*, *S.A.*).
- 2021: Implementação de SGI (Sistema de Gestão Integrado) segundo as normas ISO 9001 e ISO 27001; Construção e inauguração das instalações da *VOID Academy*.
- 2022: Inicio da atividade da *VOID Academy*.

1.2.3. Visão, missão e valores

A missão da VOID Software passa pelo desenvolvimento de produtos de *software* de alto nível de forma a responder a novos desafios, utilizando equipas *agile* de elevado desempenho que apresentam resultados de excelência e que desafiam todos os dias os limites das possibilidades.

A VOID Software tem como visão o foco em mercados tecnológicos maduros, a formação e retenção de recursos altamente qualificados em áreas tecnológicas de grande relevância, a parceria com a comunidade científica e a contribuição para o desenvolvimento do ecossistema tecnológico e empreendedor local.

A *VOID SOFTWARE* possui como valores principais:

- Pessoas antes do negócio
- Compromisso dentro da equipa e perante os clientes
- Construção contínua do conhecimento através da experiência
- Orgulho e excelência em tudo o que fazem

1.2.4. Futuros desafios

A *VOID SOFTWARE* terá como principais desafios a entrada e expansão em mercados noutras localizações geográficas, manter-se atualizada no que toca às tecnologias de programação e às outras tecnologias consideradas de elevada importância para a empresa, adotando as metodologias e técnicas mais atuais, de modo a acompanhar as necessidades dos vários mercados onde a empresa se insere.

1.3.Planeamento

O estágio curricular teve uma duração aproximada de 6 meses e meio. Foi elaborado um diagrama de *Gantt*, que se encontra representado na Figura 2, para garantir um controlo temporal sob a carga laboral durante o período de estágio.

	2023				2024		
Tarefa	Setembro	Outubro	Novembro	Dezembro	Janeiro	Fevereiro	Março
Visão geral do processo de desenvolvimento e práticas de segurança utilizadas dentro da empresa							
Visão geral das tecnologias e ferramentas utilizadas (*pipeline* CI/CD)							
Levantamento do estado de arte sobre visualização de dados em contexto de DevSecOps e segurança de aplicações							
Identificar fontes essenciais de dados para o *dashboard*							
Extração de dados das fontes previamente identificadas							
Desenho de modelo de dados							
Criação da base de dados e importação dos dados							
Desenho e desenvolvimento do *dashboard*							
Validação da solução							
Escrita do Relatório de Estágio							

Figura 2 – Diagrama de *Gantt*

1.4. Comunicações e Publicação

O trabalho desenvolvido durante o estágio curricular resultou em duas participações em eventos e na publicação de um artigo científico. A participação, como orador, na 1ª edição do *UptoData* [9] — Encontro Nacional de Estudantes de Ciência de Dados, realizado entre os dias 3 e 6 de março de 2024, em Coimbra, proporcionou a oportunidade de divulgar os resultados do trabalho desenvolvido junto da comunidade académica.

O artigo intitulado "*Security Insights at a Glance: An Integrated Data Approach for Application Security*" [10] foi publicado e apresentado na conferência "*The 18th IEEE International Conference on Application of Information and Communication Technologies (AICT2024)*" [11], decorrida entre os dias 25 e 27 de setembro de 2024, em Turim, Itália.

A publicação pode ser encontrada neste documento, no Anexo B.

1.5. Estrutura do Documento

O presente documento está estruturado em vários capítulos. De seguida é feita uma breve descrição de cada um:

O Capítulo 2 apresenta uma visão geral dos conceitos teóricos relacionados com a temática do estágio e que servem de base para o trabalho realizado.

O Capítulo 3 apresenta uma análise do trabalho existente relacionado com o tema do estágio, onde é detalhado todo o processo de revisão sistemática da literatura efetuado e apresentado os resultados mais relevantes no domínio de estudo.

O Capítulo 4 descreve as ferramentas e tecnologias essenciais utilizadas na execução das várias atividades que compõem o estágio curricular.

O Capítulo 5 aborda a metodologia aplicada no desenvolvimento da solução. As secções iniciais apresentam a arquitetura proposta, os requisitos funcionais e não funcionais, bem como a caracterização dos perfis de utilizadores para a qual a solução foi projetada. Da secção 5.4 à secção 5.8 é pormenorizada cada uma das etapas principais do desenvolvimento da solução. A secção 5.9 apresenta os diferentes componentes visuais que constituem a solução de visualização. A secção 5.10 descreve o método implementado para a validação da solução e apresenta os respetivos resultados.

No Capítulo 6 é efetuada uma análise crítica da solução desenvolvida, considerando duas abordagens. A primeira consiste numa avaliação global da solução, destacando os seus pontos fortes, bem como possíveis pontos de melhoria. A segunda abordagem consiste numa análise comparativa entre a solução desenvolvida e as soluções já existentes e previamente identificadas no Capítulo 3.

Por fim, no Capítulo 7 são apresentadas as conclusões decorrentes do trabalho de estágio desenvolvido e propostas possíveis medidas a implementar futuramente, com vista ao aperfeiçoamento da solução.

2. Estudo preliminar

Neste capítulo são apresentados os conceitos teóricos que servem de base para o trabalho elaborado no âmbito do tema do estágio curricular. A primeira secção aborda a temática da visualização de dados e os seus principais aspetos. A secção 2.2 fornece uma visão mais detalhada sobre as aplicações Web e são explorados os conceitos relacionados com a sua segurança.

2.1. Visualização de dados

A visualização de dados pode ser definida como um conjunto de métodos e técnicas utilizados para representar os dados de forma gráfica ou pictórica, tornando a informação mais compreensível para o público a que se destina [12, 13], contribuindo para explicar factos e suportar a tomada de decisão. A visualização de dados tem aplicações em vários domínios como a saúde, a educação, o marketing, etc.[12, 14].

Embora este conceito seja associado de forma recorrente ao mundo digital atual, a visualização de dados remonta há mais de um século, com exemplos de algumas visualizações de dados mais influentes datadas do século XIX [13].

No mundo contemporâneo, o processo de transformação digital mundial e o surgimento de *Internet of Things* (IoT) contribuíram para a enorme escalada na quantidade e velocidade de dados produzidos, bem como o aparecimento de uma grande variedade de novas fontes e tipos de dados. Dados provenientes das atividades digitais de cada pessoa (redes sociais, aplicações, etc.) são assim gerados diariamente, a uma velocidade e volume sem precedentes [12 – 15], impulsionando o aparecimento de conceitos como *big data*.

Neste mesmo período, inúmeras organizações começaram a tratar os seus dados como ativos, sendo até considerados a nova moeda económica [16], de onde pode extrair valor e por essa via retirar ganhos competitivos relativamente aos seus concorrentes de mercado [17].

2.1.1. Era da *Big Data*

Embora não exista um consenso no que diz respeito à quantidade exata e às características que a distinguem da restante *data* [18], o termo *big data*, na generalidade, refere-se a um conjunto de dados que, devido à sua grande quantidade e natureza complexa, não são possíveis de processar e armazenar por meio de métodos e *software* tradicional [13]. A era da *big data* trouxe novos desafios às várias organizações [18], que devido às suas características tiveram de desenvolver novos métodos e sistemas capazes de recolher, armazenar e processar não só um grande volume de dados, como também conseguir extrair e analisar informação de novos tipos e formatos de dados estruturados e não estruturados [13]. Por outro lado, *big data* representou novas oportunidades, na perspetiva de que, se existe um valor inerente nos dados para as empresas, o surgimento de novos pontos de recolha destes dados i.e. novas fontes, bem como maior volume de dados disponíveis, o valor potencial que se pode obter através da sua aquisição, processamento e análise é consideravelmente maior.

De facto, de acordo com um inquérito em 2013 [19], 64% das organizações já tinham ou planeavam investir em iniciativas relacionadas com *big data*, evidenciando a sua relevância estratégica no contexto organizacional. Com *big data*, as organizações possuem os dados brutos necessários para realizar análises complexas que recentemente seriam impossíveis de realizar e identificar padrões e tendências sociais de extrema relevância que outrora passariam despercebidos [13, 14].

2.1.2. Importância da visualização de dados

O propósito da visualização de dados vai muito para além do auxílio na tomada de decisão. Grandes empresas de comunicação como o *New York Times* e o *Wall Street Journal* reconhecem, há muito tempo, o valor da visualização gráfica, utilizando-a frequentemente como recurso para transmitir informação complexa para o seu público abrangente [13]. De modo idêntico, várias áreas científicas reconhecem a visualização de dados como uma abordagem valiosa e eficaz de disponibilizar e transmitir informação para a populações compostas por indivíduos com vários níveis de literacia sobre a área onde a informação se insere [13 – 15].

Com a adoção de uma cultura de *data-driven* por parte das organizações, os dados tornaram-se mais valorizados e confiáveis, ganhando uma enorme preponderância na tomada de decisão das organizações e na criação de novo conhecimento sobre o mundo [14]. A descoberta do potencial inerente de *big data* e o seu possível impacto para as organizações criaram um cenário de grande pressão, em contexto empresarial, para gerar e utilizar grandes quantidades de dados.

No entanto, mesmo os maiores e mais amplos conjuntos de dados podem tornar-se inúteis ou perder grande parte do seu valor sem uma forma adequada que extraía e processe as suas qualidades, traduzindo-a em *insights* acionáveis.

É, por isso, essencial adotar uma abordagem que permita identificar e compreender todos os aspetos do conhecimento oculto nos dados, caso contrário, existe o risco de *insights* considerados críticos para a empresa estarem presentes nos dados e não serem descobertos, tornando-se mais provável que a organização tome decisões de negócio mais fracas, ficando em desvantagem em relação a empresas concorrentes com maior capacidade de gestão e utilização de *big data* [13].

É a visualização de dados que permite que os dados sejam interpretados, constituindo um mecanismo chave de acessibilidade ao seu valor total [13]. As evidências da elevada importância da visualização de dados e as vantagens da sua utilização para as organizações são claras. Algumas dessas vantagens, já mencionadas ao longo desta secção e ilustradas na Figura 3, incluem [20]:

- **Compreensão de dados complexos:** A visualização de dados permite simplificar grandes conjuntos de dados, através da sua conversão em representações visuais, de mais fácil compreensão e interpretação para o ser humano.

- **Melhoria da tomada de decisão:** A visualização de dados desempenha um papel fulcral no suporte da tomada de decisão. Através da apresentação de dados de forma gráfica e simplificada, as pessoas responsáveis por tomar decisões podem rapidamente compreender os pontos-chave e assim tomar decisões mais informadas e mais precisas.

- **Comunicação efetiva de *insights*:** A visualização de dados é também uma ferramenta de *storytelling* que permite comunicar *insights* a diferentes tipos de público. Ao criar representações visuais apelativas, garante que a mensagem-alvo

não apenas seja clara e acessível, mas também facilmente compreendida e memorizada.

- **Identificação de padrões e tendências:** A representação gráfica de dados em bruto facilita a deteção de padrões e tendências existentes nos dados. Após a sua identificação, a organização pode tirar partido e utilizar estes padrões e tendências para criar impacto no seu negócio e ganhar vantagem sobre os seus concorrentes.

Figura 3 – Importância da visualização de dados, adaptado de [20]

A visualização de dados dá resposta a cada um dos desafios específicos de cada uma das vertentes que caracterizam *big data*, também conhecidas como os 3 Vs: volume, velocidade e variedade.

Focando-nos primeiramente no volume, é facilmente percetível que a enorme quantidade de informação gerada em *big data* seria impossível de processar sem recurso à representação e interpretação visual dos dados.

Já no que toca velocidade, quando os dados são gerados e armazenados rapidamente, torna-se inviável a execução manual de tarefas no processo de análise, permitindo, no limite, a análise de dados em tempo real. A visualização de dados garante uma estrutura de dados que pode ser rapidamente atualizada e modificada à medida que novos dados são disponibilizados, reduzindo o período temporal entre a recolha de dados e a orientação da tomada de decisão [13].

Os desafios associados à grande variedade de dados são abordados com recurso a variadas técnicas de visualização que permitem integrar diferentes fontes de dados, estruturados e não estruturados, fornecendo um alinhamento visual comum para a sua interpretação.

À medida que a capacidade de tratamento de grandes quantidades de dados cresce, aumenta a necessidade de respostas mais rápidas. Neste contexto, a capacidade de encontrar as melhores respostas aos diferentes cenários é primordial para estar na linha da frente e poder responder adequadamente aos desafios que se colocam a cada momento.

A resposta em tempo real é o limite e a antecipação, a linha de conduta que permite ter maior eficácia em contexto organizacional. O processo de obtenção dessa resposta passa, de forma inequívoca, pela visualização de dados.

2.1.3. Tipos de visualização

Existe um vasto leque de abordagens e técnicas de visualização de dados. A decisão do tipo de visualização mais apropriado deve ter em consideração a resposta a duas questões principais: qual é a mensagem final que se pretende comunicar e qual o público-alvo dessa mesma informação [21]. Deve ainda ter em conta o tipo de dados que se pretende exibir visualmente [22]. Existem diferentes classificações das técnicas de visualização de dados. Um estudo anterior propôs agrupar estes métodos em 5 categorias [23]:

- Comparação de categorias;
- Avaliação de hierarquias ou relações de parte para o todo;
- Exibição de mudanças ao longo de período temporal;
- Ilustração de relações e conexões;
- Mapeamento de dados geoespaciais;

Estas categorias incluem os mais comuns e populares tipos de gráficos como *tables*, *charts*, *graphs*, *maps* e *diagrams*. Na Figura 4 encontram-se representadas alguns dos tipos de gráficos mais comuns com especificação dos casos de aplicabilidade de cada.

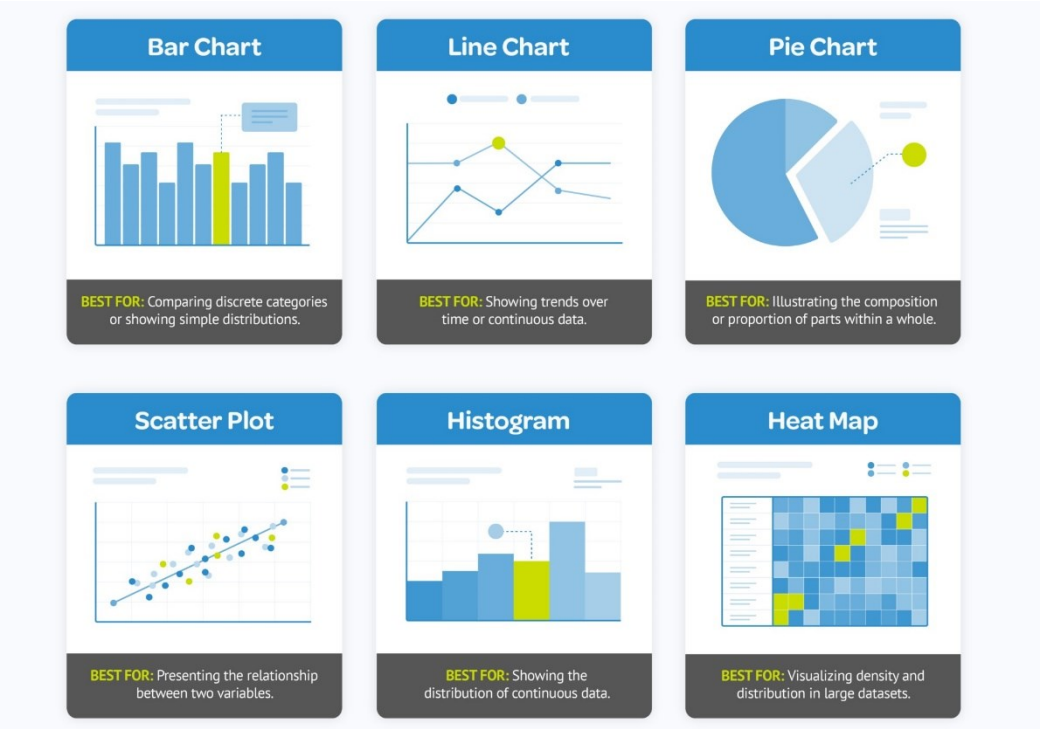


Figura 4 – Tipos comuns gráficos para *display* de dados, adaptado de [24]

O *bar chart* enquadra-se na 1ª categoria mencionada e é o exemplo ideal para comparar categorias discretas ou para representar pequenas distribuições. Um exemplo da sua utilização é na comparação dos valores de vendas entre várias categorias de produtos.

O *line chart* é útil na exibição de dados contínuos e na representação de tendências ao longo de um período temporal. Monitorização de variáveis ambientais como temperatura e humidade ou a representação do crescimento populacional são exemplos da utilização deste gráfico.

O *pie chart* permite ilustrar a composição ou proporção das partes relativamente ao todo. Pode ser utilizado para representar o dinheiro investido nos vários tipos de produtos financeiros (ações, obrigações, criptomoedas, etc.) em relação ao montante total investido.

O *scatter plot* é um gráfico de duas dimensões que permite ilustrar a relação entre duas variáveis. Um exemplo popular da utilização deste gráfico é a representação da relação entre o peso e a altura.

O histograma é utilizado para representar a distribuição de dados contínuos. Para isso, os dados são agrupados em classes de igual amplitude, cujo número é configurável e definido

pelo responsável pela sua elaboração. A representação da distribuição do rendimento da população é um exemplo da utilização deste gráfico.

O *heatmap* usa um gradiente de cores para representar os valores dos dados, num formato de matriz. São eficazes na visualização de grandes conjuntos de dados, como a interação entre utilizadores em *websites*.

Cada tipo de gráfico possui um conjunto de elementos que lhe estão associados como a posição, elementos iconográficos, cor, densidade e tamanho. São estes elementos que caracterizam a visualização [13] e determinam a sua efetividade na transmissão de informação. Escolher a técnica de visualização a utilizar e a definição dos seus elementos é assim fundamental para permitir uma análise visual exata e precisa dos dados.

2.1.4. *Frameworks* para o desenvolvimento de *dashboard*

Com o aumento da quantidade disponível de dados e o seu peso para a tomada de decisão nas organizações cada vez mais determinante, a pressão que as organizações enfrentam para incorporar metodologias que permitam uma análise de dados adequada que auxilie a tomada de decisão é cada vez maior [25]. O acesso a uma maior quantidade de dados não se traduz, necessariamente, numa melhor tomada de decisão, é necessário realizar uma análise de dados [25].

A criação e utilização de *dashboards* tem-se tornado uma prática cada vez mais popular dentro das organizações, como mecanismo de apoio útil à tomada de decisão [26]. Na sua definição mais conhecida, o *dashboard* é "*a visual display of the most important information needed to achieve one or more objectives, consolidated and organized on a single screen so the information can be monitored at a glance*" [27].

Pode-se então considerar que um *dashboard* é meio de visualização de informação mais relevante de forma que esta possa ser utilizada no auxílio da tomada de decisão. O *dashboard* é utilizado nas mais diversas áreas, como a saúde, educação, logística e o seu propósito é diversificado, baseado nas tarefas que o seu utilizador pretende realizar e nos objetivos que pretende atingir [26].

Alguns autores classificam os *dashboards* em Tático, Operacional e Estratégico, em concordância com os vários níveis de gestão existentes numa organização [26, 28]. Cada tipo de *dashboard* tem um propósito e objetivo específico, sendo que a informação apresentada difere consoante o tipo de *dashboard*.

É importante identificar corretamente o tipo e propósito do *dashboard*, tendo em conta o utilizador final e o papel que este desempenha dentro da organização, antes de se avançar para o processo de desenvolvimento e *design* do *dashboard* [26, 29].

No que concerne ao desenvolvimento de *dashboard*s, são várias as abordagens adotadas pelos autores de trabalhos publicados. Um estudo recente visou incorporar os princípios da interação humano-computador (HCI) no desenvolvimento de *dashboards* de manutenção focados no utilizador [30]. Outro estudo compilou uma lista de 16 princípios de *design* de *dashboard* no contexto de dados públicos governamentais, através de uma revisão sistemática da literatura [31]. No contexto da cibersegurança, uma publicação recente propôs uma *framework* para visualização e análise de dados em tempo real [32].

Embora estes *frameworks* para desenvolvimento e *design* de *dashboards* apresentem diferenças entre si, seguem um conjunto de etapas principais, que são detalhadas em seguida.

Planeamento e levantamento de requisitos

Nesta etapa é definido o propósito e objetivo do *dashboard*, quais os utilizadores finais e/ou audiência e quais as métricas e KPIs que necessitam de estar presentes [33, 34]. É necessário entender quais as questões a que o *dashboard* pretende dar resposta e o tipo de utilizador base, de forma a personalizar o *dashboard* de acordo com as suas necessidades [34].

Recolha e preparação de dados

Nesta etapa são identificadas as fontes de dados necessárias. Após esta identificação inicial, os dados são devidamente extraídos e pré-processados, através de um processo ETL [33, 34]. Este processo visa garantir que os dados são precisos e consistentes, garantindo que a análise da informação presente seja confiável.

Design do *dashboard*

Nesta fase é realizado um esboço e prototipagem do *dashboard*, com recurso a *mockups* para definir a sua estrutura e aspeto. Pode ser discutido e definido com as partes interessadas os elementos fundamentais que irão constituir o *dashboard* como os tipos de gráficos a utilizar, a sua localização. Este processo iterativo de *design* e *feedback* permite validar o aspeto do *dashboard* junto dos utilizadores antes do início do seu desenvolvimento [34].

Desenvolvimento do *dashboard*

A fase seguinte consiste no desenvolvimento concreto do *dashboard*. O desenvolvimento deve considerar as boas práticas de visualização de dados [31], como a utilização de tipo de gráfico adequado, escolha de um esquema de cores apropriado, garantir um suporte interativo ao utilizador, permitindo a aplicação de filtros e *drill-downs* [34, 35].

Testes e Validação

Concluído o desenvolvimento procede-se à realização de testes de forma a verificar se o *dashboard* cumpre os requisitos propostos inicialmente. Podem ser realizados testes de precisão de dados representados, onde é verificado se os dados presentes no *dashboard* correspondem aos dados presentes na base de dados ou testes com utilizadores, em que uma amostra representativa dos tipos de utilizadores alvo testa as funcionalidades da plataforma ou testes de performance [34].

Partilha e documentação

Após a sua validação, o *dashboard* é disponibilizado aos utilizadores, com toda a documentação necessária, como tutoriais, para garantir a sua correta utilização [33, 34].

Monitorização e melhoria contínua

A fase final consiste na monitorização contínua do *dashboard*, por meio de métricas de performance e de adoção por parte dos utilizadores (número de utilizadores ativos, dias desde a última visualização, etc.) e na implementação de um mecanismo de recolha de *feedback* dos utilizadores que será utilizado para fins de otimização do *dashboard* [33, 36].

2.1.5. Ferramentas *low-code* de visualização de dados

A falta de programadores profissionais de *software* no mercado de trabalho leva a que muitas organizações enfrentem dificuldades na sua transformação digital. Além do mais, muitos projetos de desenvolvimento de *software* apresentam frequentemente problemas de eficiência ou falham por completo [37].

Nos últimos anos surgiu um novo tipo de ambiente de desenvolvimento de *software*, designado de *low-code*. Este tipo de solução tem ganho visibilidade e a tornar-se uma tendência, tanto no mundo empresarial como na comunidade académica [37]. Com o aumento de popularidade do termo "*low-code*", grandes empresas de tecnologia com a IBM e a Microsoft começaram a integrar este tipo de soluções no seu portefólio de produtos e a vender as plataformas *low-code* (LCP) como uma possível solução para os problemas mencionados [37].

Segundo a IBM [38] "Quando consegue criar visualmente novas aplicações empresariais como o mínimo de programação manual – quando os seus programadores conseguem algo com maior valor, mais rapidamente – isso é *low code*".

O surgimento das LCP também levou a uma mudança de paradigma no domínio da visualização de dados, uma vez que a abordagem tradicional na criação de representações visuais de dados muitas vezes carece de competências de programação [39], criando possíveis entraves em muitas organizações no acesso aos *insights* presentes nos dados.

Embora não exista uma definição formal e distinta destes sistemas [40] existe um conjunto de características e funcionalidades que estão frequentemente presentes nas plataformas *low-code* [39, 40], entre as quais:

- **Ambiente visual de desenvolvimento:** A presença de funcionalidades *drag-and-drop* possibilitam o desenvolvimento de aplicações de forma visual.
- **Modelos e componentes pré-definidos:** Estas plataformas costuma oferecer um conjunto de elementos prontos a usar que podem ser customizados de acordo com as necessidades específicas.
- **Conetores embutidos:** Muitas LCP possuem conetores que permitem integrar facilmente a plataforma com diversas fontes de dados e serviços de terceiros.
- **Geração automática de código:** A plataforma traduz os vários painéis de visualização em código funcional, no *background*, o que proporciona ao utilizador criar visualizações complexas sem ser necessário recorrer à escrita de código.
- **Pré-visualização instantânea:** Alterações são visualizadas em tempo real.

Esta tecnologia apresenta diversas vantagens, comparativamente com as abordagens tradicionais de programação [39]. Uma das vantagens mencionadas é a redução do tempo de desenvolvimento, uma vez que os projetos são desenvolvidos mais rapidamente.

Há uma redução das barreiras técnicas, já que os utilizadores não necessitam de um conhecimento técnico extenso de linguagens de programação para criar visualizações complexas.

A utilização destas plataformas reduz ainda a necessidade da contratação de quadros mais especializados, levando redução nos custos de desenvolvimento dos projetos.

Na área da visualização de dados, a emergência das plataformas *low-code* desencadearam um processo de democratização de acesso aos dados. Este processo levou a um incremento geral da literacia de dados nas organizações [39].

A utilização desta tecnologia por colaboradores menos técnicos capacitou-os para desenvolver visualizações complexas, reduzindo a dependência das organizações em equipas especializadas de *business intelligence* (BI) ou de TI para o desenvolvimento de relatórios ou *dashboards*. O foco destas equipas passa, deste modo, a estar na realização de tarefas mais complexas, o que se traduz num aumento produtividade da organização. A

democratização de acesso aos dados, conjugado com a capacitação de colaboradores não técnicos e a redução do apoio em equipas especializadas leva a uma produção mais rápida de *dashboards* e *insights*, permitindo uma análise de dados e tomada de decisão em tempo real, garantindo maior rapidez de resposta às mudanças de mercado e/ou às necessidades dos seus clientes [39].

Para além das vantagens mencionadas, as LCP trouxeram ainda outros benefícios organizacionais, como o aumento da colaboração entre equipas técnicas e não técnicas e a promoção de uma cultura de *data-driven* na organização.

Embora ofereçam diversos benefícios, as LCP de visualização de dados ainda possuem limitações e desafios. A facilidade de utilização destas plataformas permite que a organização disponibilize o seu uso a vários colaboradores, o que pode originar problemas no âmbito da segurança, proteção e privacidade dos dados, caso esses utilizadores tenham acesso indevido a dados sensíveis, para os quais não possuem a devida autorização [39]. Podem também surgir questões de conformidade com os regulamentos em vigor, que pode ter como consequência a aplicação de multas avultadas à organização.

Estas plataformas apresentam também problemas de escalabilidade, onde a sua performance diminui ao lidar com *datasets* maiores e no cálculo de KPIs mais complexos, o que resulta numa capacidade mais limitada em representar estratégias de negócio mais sofisticadas [39].

Caso não haja uma definição clara dos dados, métricas e KPIs na organização, o uso destas plataformas pode levar à criação de silos de dados, através da criação de visualizações incoerentes por parte dos diferentes colaboradores e/ou departamentos. A inexistência de uma clara governança de dados pode levar à falta de uma visão única e central dos dados, com o possível aparecimento de várias “fontes de verdade” na organização [39].

A utilização dos dados por múltiplos colaboradores pode levar a problemas de linhagem de dados, onde se torna difícil de efetuar o rastreamento dos dados originais e todas as transformações efetuadas [39].

2.2. Web applications

Uma vez que os dados utilizados no plano de estágio estão relacionados com vulnerabilidades existentes em aplicações Web (*Web Apps*), é explicada a sua origem e modo de funcionamento.

2.2.1. Definição e arquitetura

O aparecimento da *Internet* permitiu a partilha de informação a nível mundial, impulsionando o desenvolvimento de várias tecnologias *Web.* A evolução desta tecnologia levou ao surgimento das *Web Apps* [41]. Estas aplicações são caracterizadas pela sua portabilidade e compatibilidade multiplataforma [42].

Contrariamente às aplicações tradicionais, que só podem ser utilizadas no dispositivo onde se encontram instaladas, as *Web Apps* são executadas num *browser*, permitindo o seu uso de forma eficiente em qualquer dispositivo, desde que sejam devidamente responsivas. Além disso, de forma geral a sua utilização é independente do sistema operativo do dispositivo onde é executada, sendo apenas necessário que este consiga aceder à *Internet* [42, 43].

Como representado na Figura 5*,* a *web application* usualmente assenta numa arquitetura cliente-servidor, dividida concetualmente por 3 camadas [44]:

- ***Presentation/ Client Layer:*** Camada visível para os utilizadores, e que garante a interação com o sistema, podendo ser realizada através do *browser,* de um aplicativo móvel ou de uma aplicação *desktop*. É responsável por receber os pedidos de utilizadores, apresentando a informação necessária ou realizando as ações solicitadas.

- ***Application/ Business Layer:*** Camada responsável por processar os pedidos, implementar a lógica de negócio e disponibilizar os dados à camada de apresentação. É, adicionalmente, responsável por comunicar com a camada de dados (*Data Layer*).

- ***Data Layer:*** Camada responsável pelo armazenamento e gestão dos dados do sistema.

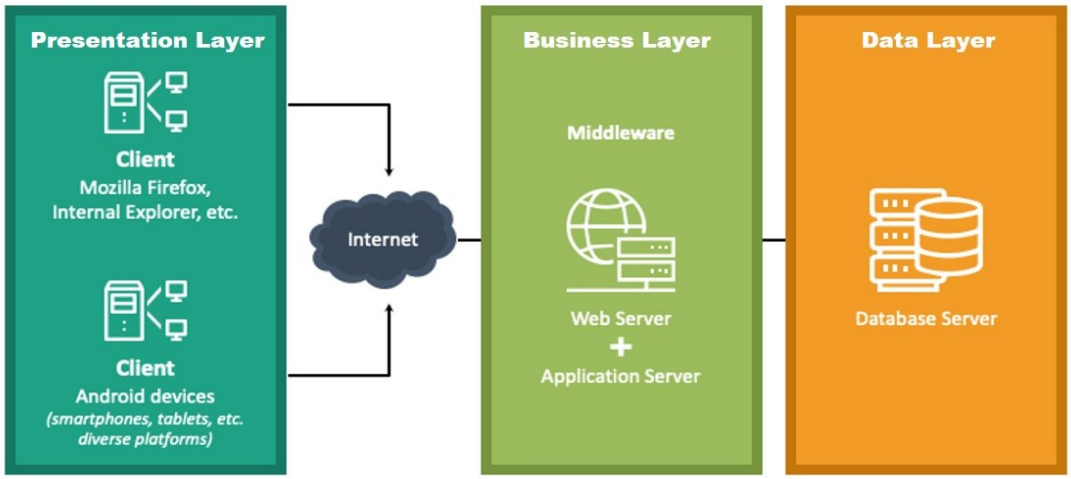


Figura 5 – Arquitetura das *Web Applications*, adaptado de [45]

2.2.2. Vulnerabilidades de aplicações Web

Vulnerabilidade, no sentido lato, significa um ponto mais frágil de algo ou alguém, podendo servir como alvo de ataque. No âmbito das aplicações Web, vulnerabilidades referem-se a defeitos detetados no *software* desenvolvido e que podem ser explorados por terceiros, resultando em quebras de segurança e levando a perdas ou prejuízos para a organização [46]. A utilização massiva por parte de utilizadores de dispositivos e aplicações móveis traduz-se numa exposição mais acentuada a possíveis vulnerabilidades [47].

As vulnerabilidades podem ser divididas em vários tipos, consoante as suas características e podem ser encontradas nas 3 camadas das aplicações Web. Uma das organizações mais conhecidas, a *Open Web Application Security Project* (OWASP), lança periodicamente uma lista dos 10 riscos de segurança mais comuns nas aplicações denominada de OWASP - *Top Ten* [48]. À data atual a lista é composta pelas seguintes vulnerabilidades:

- A01: Broken Access Control - Falha no controlo de acesso que garante a um ou mais utilizadores terem privilégios de acesso indevido a informação e/ou conteúdo;
- A02: Cryptographic Failures - Falhas relacionadas com a criptografia ou a sua ausência, podendo levar a exposição de dados sensíveis;
- A03: Injection - Injeção de dados ou comandos maliciosos;

- A04: Insecure Design - Lacunas existentes no *design* de segurança implementado, levando a um controlo de segurança ineficaz;
- A05: Security Misconfiguration - Exploração de configurações de segurança incorretamente definidas;
- A06: Vulnerable and Outdated Components - Utilização de dependências desatualizadas e/ou com vulnerabilidades conhecidas no desenvolvimento das aplicações;
- A07: Identification and Authentication Failures - Falhas no mecanismo de identificação e autenticação dos utilizadores, permitindo aos atacantes assumir a identidade de outros utilizadores de forma temporária ou permanente;
- A08: Software and Data Integrity Failures - Falhas na verificação da integridade dos dados e *software* utilizado. Um exemplo desta vulnerabilidade é quando aplicação utiliza *plug-ins* e módulos provenientes de fontes não fidedignas;
- A09: Security Logging and Monitoring Failures - Procedimentos de monitorização, registo de eventos e medidas de resposta de segurança inadequados, insuficientes ou inexistentes, impossibilitando assim a identificação das vulnerabilidades existentes;
- A10: Server-Side Request Forgery (SSRF) - Falha em que o utilizador tenta aceder a dados e recursos da aplicação remotamente, sem a necessidade de validar o URL fornecido por este;

2.2.3. *Continuous Integration & Continuous Delivery*

A evolução das *Web Apps* contribuiu para a mudança no foco das empresas em desenvolver *software* como um serviço (SaaS) em vez de como um produto (SaaP). A supramencionada alteração permite que as equipas de desenvolvimento de *software* melhorem de forma contínua as suas aplicações, e consequentemente os seus serviços. A disponibilização de novas versões com novas funcionalidades passa a ser mais ágil, melhorando os índices de satisfação dos clientes [5, 49].

Nesse sentido, é enunciado um conjunto de práticas designadas de *Continuous Integration & Continuous Delivery* (CI/CD) que possibilita às empresas alcançar esse

ambiente de desenvolvimento de *software* ágil, sem menosprezar os padrões de qualidade e conformidade obrigatórios.

Num ambiente de *Continuous Integration* (CI) a equipa de desenvolvimento produz pequenas alterações no código de uma aplicação e integra-as num repositório de versionamento de código (como, por exemplo, o Git) [49] controlando desta forma a integração do histórico de codificação realizado [50 – 52].

Em *Continuous Delivery* (CD) o *deploy* de novas versões é realizado de forma frequente, mas garantindo índices de qualidade elevada. As novas versões podem incluir, entre outras, novas funcionalidades e correção de erros. A Figura 6 representa um exemplo genérico de uma *pipeline* CI/CD.

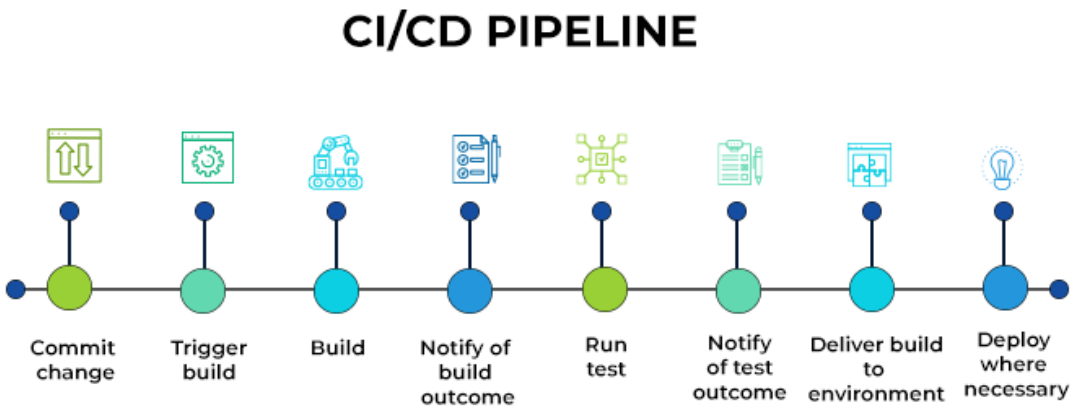


Figura 6 – Exemplo de um *pipeline* CI/CD, retirado de [53]

As práticas de CI/CD são suportadas por ferramentas que permitem a automatização de vários processos, através da criação de um *pipeline* de CI/CD. No capítulo 4 é explicada qual a ferramenta utilizada pela *VOID SOFTWARE* neste contexto.

2.2.4. DevSecOps

A implementação das práticas estabelecidas em CI/CD levantou inicialmente vários desafios às organizações, uma vez que tanto a equipa de desenvolvimento como a equipa operacional realizavam as suas tarefas de forma independente e isolada [5]. O conceito inicial, designado

de DevOps, foi criado para ultrapassar estes desafios e tinha como base a colaboração e partilha de responsabilidades entre as duas equipas em todas as etapas do ciclo de vida do desenvolvimento de *software* (SDLC), realizando ações conjuntas de resolução de problemas, definição de métricas e automatização de processos [5, 50, 51].

Recentemente, o aumento do cibercrime levou a grandes prejuízos por parte das empresas, com a violação e roubo de dados a custar em média 3.86 milhões de dólares em 2020 [54]. Este fenómeno em conjunto com estabelecimento de regulamentos de segurança e privacidade, como o Regulamento Geral da Proteção de Dados (RGPD) a nível de UE [51], levou a uma crescente preocupação por parte das organizações na garantia de segurança das suas aplicações e em estar em conformidade com os regulamentos em vigor.

Nasceu, assim, o conceito de DevSecOps, que alia as práticas de segurança com as práticas já estabelecidas de DevOps, estando presente em todo o SDLC. Isto permite identificar vulnerabilidades em fases iniciais do ciclo, sendo mais facilmente corrigidas do que se fossem encontradas em estágios de desenvolvimento mais avançados [47]. Um dos focos de DevSecOps é a automatização de testes de segurança.

A Figura 7 ilustra as práticas de DevSecOps em todas as etapas do SDLC. As práticas de DevSecOps são frequentemente representadas pelo símbolo de infinito, salientando a sua natureza contínua no SDLC. As várias práticas e processos desenvolvimento, segurança e operações são implementadas ao longo de fluxo contínuo, que se inicia na fase embrionária das aplicações em desenvolvimento, passando pela implementação (*deploy*), até à sua monitorização. Este ciclo ininterrupto garante uma segurança e melhoria contínua das aplicações, sem comprometer a agilidade na disponibilização de novas versões aos utilizadores [6].

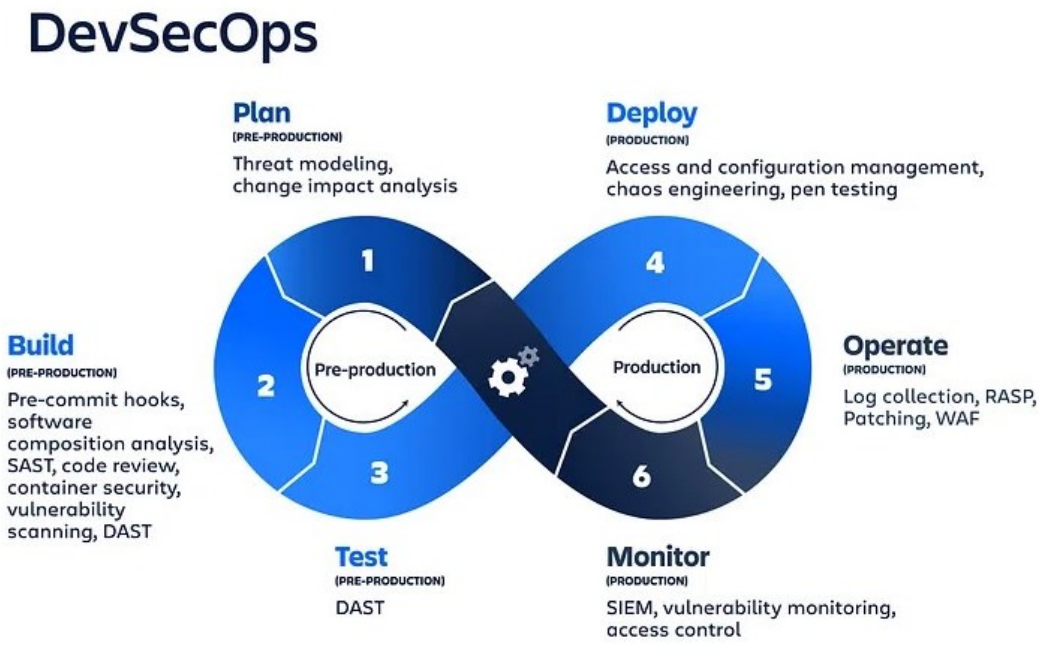


Figura 7 – Práticas de DevSecOps, retirado de [6]

3. Visualização de dados em segurança de aplicações

Neste capítulo são abordados os estudos realizados e o conhecimento atual inerente à área de visualização de dados em segurança de aplicações. A primeira secção descreve a metodologia aplicada no processo de revisão sistemática de literatura. A secção seguinte detalha os trabalhos previamente elaborados, as arquiteturas e soluções existentes dentro da temática de trabalho. Na secção final é feita uma análise crítica dos trabalhos descritos.

3.1. Metodologia da revisão sistemática de literatura

Através da realização da Revisão Sistemática de Literatura (RSL) procedeu-se à recolha e análise dos estudos mais relevantes da área na qual o tema de estágio se insere. Todo o processo seguiu as normas do método *Preferred Reporting Items for Systematic Reviews and Meta-Analysis* (PRISMA) [55]. Este processo é representado na Figura 8, através de um diagrama de fluxo. Como é possível observar, este método encontra-se dividido em três fases distintas. A primeira fase é a identificação que consiste na pesquisa e obtenção dos trabalhos relativos ao tema. A fase seguinte designa-se de *screening*, onde os artigos encontrados são filtrados e selecionados conforme a sua relevância no contexto da temática em estudo. A última etapa regista o número de publicações incluídas na revisão de literatura. Para além da divisão por fases, o diagrama também se divide em 2 colunas, a primeira menciona os registos encontrados em bases de dados de carácter académico-científico e a segunda a registos encontrados por outras vias.

Na fase de identificação, a pesquisa de publicações foi realizada em quatro bases de dados distintas, utilizadas amplamente no meio académico, que constituem um grande repositório de registos revistos e validados cientificamente: *Google Scholar* [56], *ACM Digital Library* [57], *IEEE Xplore* [58] e *Springer* [59]. Foram utilizados os seguintes termos na pesquisa: "*application security*" AND *vulnerability* AND (*dashboard* OR "*data visualization*").

A pesquisa inicial resultou na identificação de um total de 683 registos, no conjunto das quatro bases de dados utilizadas. Devido à quantidade de publicações ser demasiado elevada para prosseguir para a etapa seguinte do processo de revisão, utilizaram-se filtros de forma

a restringir o número de registos. Desta forma, foram apenas considerados os registos com até cinco anos desde a data de publicação (> ou = 2019) cujo idioma de escrita fosse inglês, tendo os restantes artigos sido removidos da análise. Uma vez que o número de publicações restantes permanecia muito elevado, filtrou-se os resultados por relevância, considerando somente os 20 primeiros artigos de cada base de dados. Os resultados dessa filtragem foram posteriormente agrupados numa única lista. Nesse agrupamento, verificou-se a presença de registos em duplicado, que foram removidos. No final da fase de identificação, um total de 45 publicações passaram para a fase de *screening*.

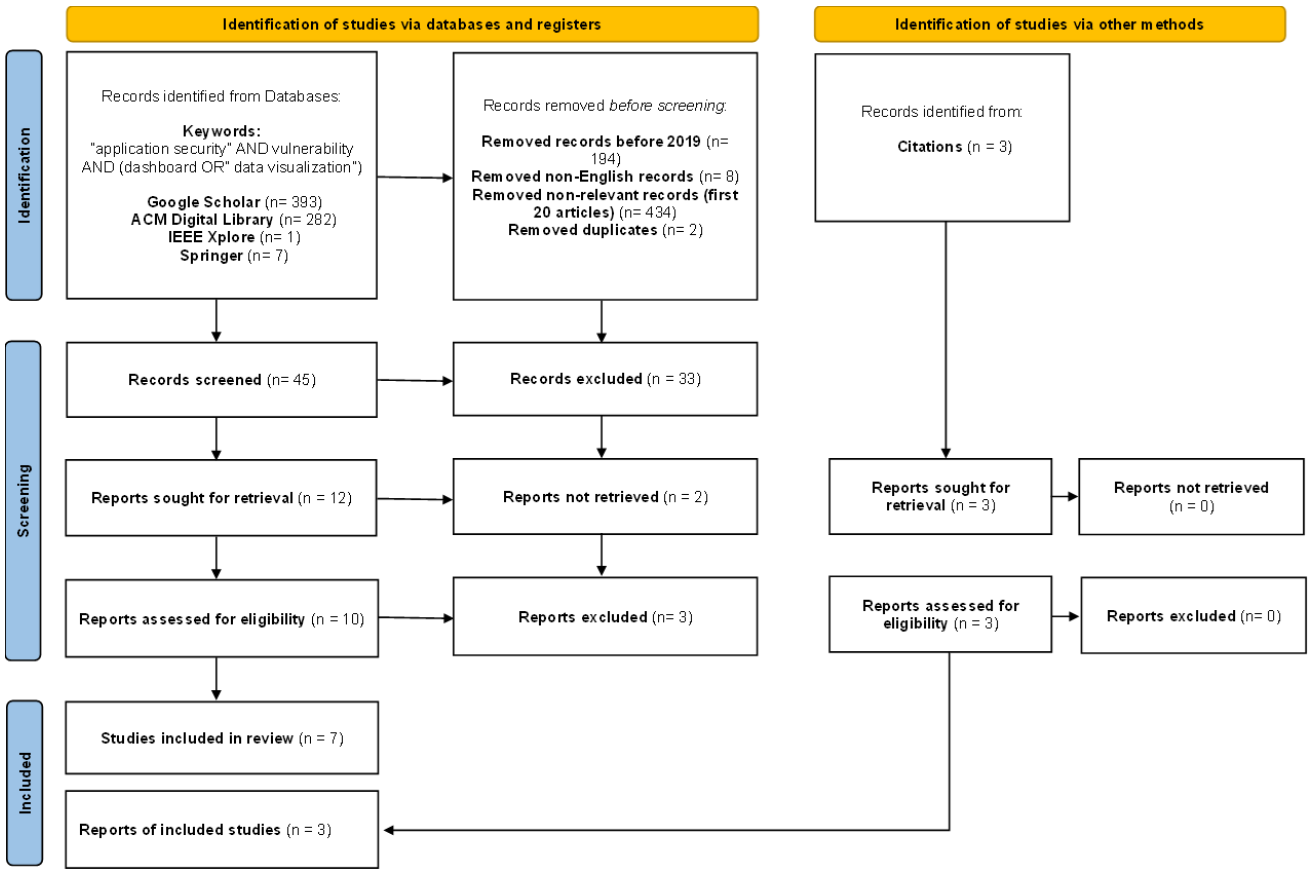


Figura 8 – Diagrama de fluxo do processo de revisão de literatura

Nesta 2ª fase do processo, verificou-se a potencial relevância das publicações consideradas, através da análise do seu título e *abstract*. De mencionar que esta avaliação foi efetuada em conjunto com um colaborador da entidade de acolhimento do estágio, para além do autor do presente documento. De forma a efetuar a análise, utilizou-se uma folha de

cálculo, que se encontra representada no Anexo C, para criar uma tabela comparativa, com três colunas, uma contendo o título da publicação, outra com o resumo e uma coluna com a avaliação atribuída à publicação, considerando as duas primeiras colunas, numa escala de 0 a 2 valores. Este método de avaliação foi utilizado por ambos os intervenientes, de forma independente, assegurando que não existia influência de interveniente para com o outro. Foi definido previamente que registos cujo valor obtido da soma das avaliações fosse igual ou superior a 2 seriam considerados suficientemente relevantes para uma análise mais aprofundada. Como resultado desta fase, foram selecionados 12 artigos, tendo os 33 restantes sido descartados.

No passo seguinte do processo, tentou-se obter o conteúdo integral dos registos ainda em análise, avaliando de forma mais profunda o relevo deste conteúdo para o trabalho em causa. Duas publicações foram excluídas devido ao seu conteúdo integral não estar acessível gratuitamente. O conteúdo de cada um dos restantes artigos foi avaliado tendo em conta os seguintes critérios de aceitação: relevância, intuito da pesquisa, qualidade científica e a qualidade da escrita. Esta avaliação resultou na exclusão de mais três registos.

Para além do procedimento regular, o PRISMA incorpora um conceito adicional para incluir outras publicações consideradas relevantes para a revisão da literatura e que não foram originalmente identificados através da metodologia regular de pesquisa. Por esta metodologia foram identificadas três publicações consideradas relevantes, encontradas nas citações aquando da análise do conteúdo integral dos registos regulares.

Todo o procedimento resultou assim num total de 10 registos que foram incluídos na revisão sistemática de literatura, por se considerar que contêm informação relevante e valiosa no contexto do tema do trabalho de estágio realizado. A informação obtida destes registos é abordada na secção seguinte.

3.2. Application Security Visualization

No que toca ao domínio da *security and cybersecurity visualization*, já existe um grande conjunto de trabalhos publicados que abordam essa temática. Um estudo definiu 11 áreas de foco para avaliar o nível de maturidade da cibersegurança nas organizações [60], sendo uma das áreas mencionadas a segurança de aplicações. A aceitação do uso de um *dashboard* para

medir e avaliar o estado de maturidade de cibersegurança nas organizações foi examinada através do *Technology Acceptance Model* (TAM) [61], onde foi destacado que a combinação de *frameworks* com avaliação de maturidade podem ser realizados através da visualização com *dashboards*, facilitando a compreensão por parte dos colaboradores do estado atual da cibersegurança da organização e do seu nível de maturidade.

Uma *gap analysis* elaborada no domínio da *security visualization* [62] constatou que embora muitas áreas da segurança tenham sido abordadas, o número de estudos relativos às vulnerabilidades de segurança das aplicações web eram ainda muito limitados.

Na área de *application security* foi desenvolvido um *Web-based dashboard* para monitorizar as atividades de *pentesting* considerando os padrões da OWASP, mostrando as vulnerabilidades das aplicações encontradas por este com base na frequência de ocorrência [63].

Os estudos realizados nesta área focaram-se nos dados dos relatórios gerados pelas ferramentas de *Static Application Security Testing* (SAST) para visualizar as vulnerabilidades das aplicações. A ferramenta Cesar foi desenvolvida com o propósito de promover a colaboração entre os programadores e a exploração das vulnerabilidades, permitindo aos programadores focarem-se quer na qualidade global do código fonte do *software* desenvolvido, quer em vulnerabilidades específicas.

O Cesar foi desenvolvido como aplicação web, de forma a poder ser acedida em várias plataformas através de um *browser*, sem a necessidade de realizar qualquer instalação. Esta ferramenta utiliza um *treemap* para mostrar visualmente as vulnerabilidades do *software* identificadas pelo *scanner FindBugs* [64]. As vulnerabilidades encontram-se agrupadas em categorias (*security*, *malicious code*, *bad practice,* etc.). Os utilizadores podem selecionar quais as categorias de vulnerabilidades que são representadas no *treemap* através da *checkbox* situada acima do painel de visualização.

Outro trabalho adotou uma abordagem distinta no design da ferramenta de visualização de vulnerabilidades, com maior foco no utilizador e em funcionalidades interativas [65]. Desenvolvida para dois grupos de utilizadores, programadores de software e analistas de segurança, esta ferramenta permite carregar as aplicações *android* e a informação das vulnerabilidades identificadas pelo SAST.

Possui duas vistas principais: a primeira exibe a informação das vulnerabilidades identificadas num relatório de segurança, numa visualização em formato de matriz. A segunda vista permite comparar as vulnerabilidades entre duas versões diferentes da aplicação, através de um *unit bar chart*. Isto permite avaliar o progresso da aplicação ao nível de segurança e identificar problemas que surgiram em versões mais recentes.

Uma solução de visualização desenvolvida combinou dados de relatórios gerados por diferentes ferramentas SAST com informação da proveniência relativa ao código fonte da aplicação [66]. Os dados da proveniência são extraídos dos repositórios das aplicações, no GitLab, para formato JSON. Estes dados contêm informação relativa às diversas atividades que ocorrem durante o ciclo de desenvolvimento (*commits, releases, warnings,* etc.). Esta informação é armazenada numa base de dados de grafos, sendo depois utilizada em conjunto com a informação presente nos relatórios de vulnerabilidades para a criação do *dashboard*.

A incorporação e armazenamento da informação de proveniência no modelo de dados possibilita a filtragem de informação de acordo com eventos específicos no ciclo de desenvolvimento de *software*, como *commits*, e realizar uma análise visual dos resultados obtidos pelos SAST quando esses eventos ocorreram.

Foram ainda encontrados dois estudos, no âmbito da *application security visualization*, que se focaram nos resultados de testes de ferramentas de *Dynamic Application Security Testing* (DAST). O primeiro estudo propôs uma estrutura que utiliza não apenas resultados dos *scans* das ferramentas DAST, mas também pela informação das relações pai-filho entre os domínios URL testados e as suas respetivas páginas [67].

A arquitetura do sistema implementado permite a integração de dados de múltiplos *scanners* de segurança DAST, através da extração de atributos comuns entre os vários *scanners* e da introdução de um componente que realiza um processo de normalização individual dos resultados, transformando-os num formato comum, para serem posteriormente utilizados pelo componente de visualização de estatísticas. Neste componente as estatísticas da informação obtida são calculadas e representadas graficamente.

A solução denominada HWAS-V [7] foi desenvolvida através de uma abordagem holística, que permite a visualização os resultados obtidos pelos testes de DAST combinados com outros atributos relacionados com o projeto e a aplicação como tamanho da aplicação,

número de módulos utilizados no desenvolvimento do código, número de bibliotecas externas, etc.

Esta solução incorpora ainda aproximadamente 50 métricas e medidas diferentes nas várias vistas do *dashboard*. A informação dos padrões da OWASP, da *Web Application Security Consortium* (WASC) e da *Common Weakness Enumeration* (CWE) são armazenadas no modelo de dados proposto de forma a permitir a associação das regras de *scan* a esses mesmos padrões.

3.3. Análise Crítica

Após análise do trabalho realizado no domínio onde o trabalho de estágio se enquadra verifica-se que as soluções desenvolvidas especificamente para visualização de dados relativos às vulnerabilidades de segurança das aplicações ainda são escassas. Ainda que algumas das soluções apresentadas tivessem como abordagem, o uso de resultados provenientes de mais do que um *scanner* de segurança como fonte de dados, integrando-os com outras fontes relacionadas com a aplicação, os estudos existentes focam-se apenas na visualização de dados provenientes de um único tipo de *scanner* (SAST ou DAST), nunca combinando os resultados de vários tipos num único *dataset*, de forma a possibilitar uma análise visual integrada do estado de segurança das aplicações desenvolvidas.

Como tal e após análise cuidada, dentro do conhecimento do autor do presente relatório, não existe nenhum trabalho prévio na área com uma abordagem semelhante à adotada neste trabalho de estágio, que combina a informação dos relatórios de segurança das ferramentas SAST e DAST, bem como o uso de dados de vulnerabilidades das bibliotecas externas utilizadas pela aplicação identificadas pelos de *Software Component Analysis* (SCA) e dados resultantes dos testes automáticos de integração *end-to-end* (E2E), para fornecer aos utilizadores alvo uma visão integral e holística do estado atual de segurança das aplicações e monitorizar esse mesmo estado ao longo tempo. Com isto, o trabalho desenvolvido espera fornecer uma forte contribuição nesta área. No capítulo 6 é realizada uma análise comparativa mais detalhada, entre a solução desenvolvida neste trabalho e as restantes já mencionadas.

4. Tecnologias e Ferramentas Utilizadas

De forma que todo o trabalho realizado possa ser entendido na sua plenitude, é necessária uma explicação de todas as ferramentas envolvidas que integram a *pipeline* CI/CD, bem como outras ferramentas necessárias para a concretização das tarefas do plano de estágio. De salientar que o Jenkins, embora não faça parte do conjunto de ferramentas utilizadas na execução das tarefas de estágio, está presente neste capítulo, pois é fundamental para a implementação do *pipeline* CI/CD, ambiente no qual as ferramentas de segurança se encontram integradas.

4.1. Ferramentas de segurança

A implementação de testes automáticos de segurança através de *scanners* de segurança, é uma das práticas adotadas no contexto de DevSecOps. Após a execução dos testes, estas ferramentas geram um relatório detalhado com resultados obtidos. No âmbito do plano de estágio, esses relatórios servem como fonte de dados para a solução desenvolvida.

4.1.1. WSAP

O WSAP (*Web Security Automation Project*) foi desenvolvido com o intuito de melhorar o desenvolvimento do processo CI/CD na *framework* do *Jenkins*, permitindo a integração de diferentes ferramentas de segurança [68], garantindo assim uma camada adicional de segurança às aplicações criadas. É composta por 3 ferramentas de *scanner* de vulnerabilidade de aplicações *Web*:

- Insider CLI
- Wapiti
- OWASP ZAP

O Insider [69] é uma ferramenta que realiza testes estáticos de segurança de aplicações (SAST), criada pela InsiderSec. Após os testes, é gerado um relatório em formato JSON ou HTML com a lista de vulnerabilidades encontradas na aplicação testada, contendo ainda

informação adicional como a descrição da vulnerabilidade e respetiva classificação CVSS (caso tenham sido encontradas) [70].

Wapiti [71] é uma ferramenta *open-source* que procura vulnerabilidades de forma dinâmica (DAST), procurando formas de injetar dados na aplicação [72].

OWASP ZAP [73] é uma ferramenta *open-source* que analisa as vulnerabilidades através de testes de penetração que podem ser realizados de forma automática.

4.1.2. Cypress

Cypress é uma ferramenta que permite realizar testes de integração das aplicações Web [74]. O Cypress utiliza cenários definidos previamente, tendo em conta os requisitos da aplicação e testa esses mesmo cenários, de forma a validar se a aplicação tem o comportamento desejável, de acordo os requisitos. A este processo dá-se o nome de *Behaviour Driven Development* (BDD) [75]. Estes testes podem ser automatizados, são escritos em *JavaScript* e podem ser utilizados tanto pelos programadores da aplicação como pelos engenheiros de *quality assurance* (QA).

4.1.3. OWASP Dependency Track

OWASP Dependency Track é uma ferramenta de análise de composição de *software* (SCA) inteligente. Além de analisar e identificar vulnerabilidades nas dependências utilizadas pela aplicação, verifica também as licenças dessas dependências. A Figura 9 apresenta alguma da informação exibida pelo *dashboard* dessa ferramenta, como o número de projetos analisados, o número de vulnerabilidades total e a sua evolução ao longo do tempo, a classificação de risco atribuída, etc.

Esta ferramenta tem uma abordagem divergente das restantes ferramentas de SCA pois tira proveito da lista *Software Bill of Materials* (SBOM), garantindo assim os requisitos regulatórios e conformidade da aplicação desenvolvida [76]. Os resultados da análise podem ser visualizados através da interface gráfica desta ferramenta.

Figura 9 – Exemplo da página inicial da plataforma do OWASP Dependency Track, retirado de [77]

4.2.Jenkins

O Jenkins é uma ferramenta *open source* muito popular que foi desenvolvida com o intuito de transformar parte do ciclo de vida do desenvolvimento de *software* (SDLC) num processo de integração contínua (CI), automatizando parte das tarefas que compõem esse ciclo [78, 79]. O Jenkins permite assim a implementação de um ambiente CI/CD através da configuração e gestão de várias *pipelines*. Suporta várias linguagens de programação e possui um grande número de *plug-ins*, permitindo a integração de várias ferramentas de teste e de desenvolvimento [80]. Cada projeto possui uma *pipeline* específica. Esta *pipeline* encontra-se dividida em várias *stages,* como pode ser observado na Figura 10.

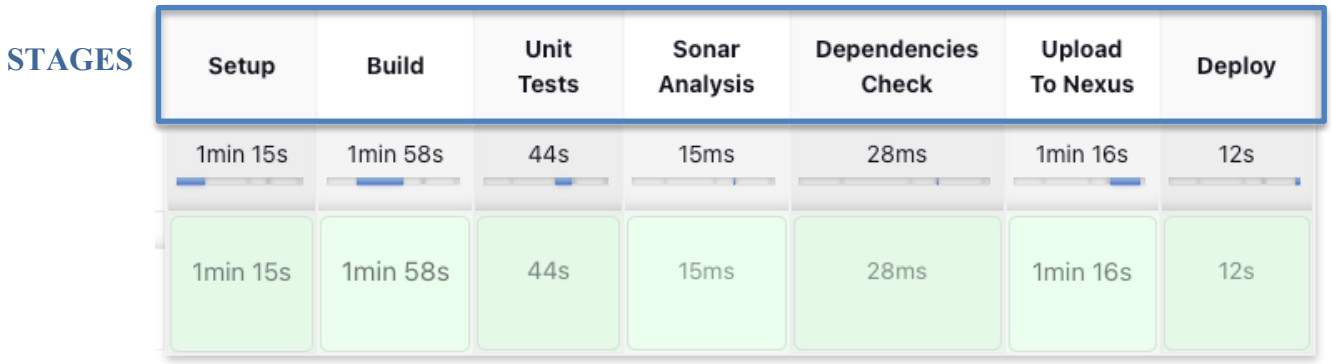


Figura 10 – Exemplo de uma *pipeline* do Jenkins

Atualmente, das ferramentas de segurança referidas, apenas o OWASP Dependency Track está inserido na *pipeline*. As restantes ferramentas são automaticamente executadas periodicamente, realizando os testes à última *build* desenvolvida da aplicação.

4.3.DBeaver

O DBeaver é uma ferramenta cliente de bases de dados que tem como propósito a gestão e administração de diferentes sistemas de bases de dados. Embora a interação entre o utilizador e as bases de dados seja maioritariamente através da sua interface gráfica (GUI), o DBeaver suporta a execução de comandos em linguagem SQL, permitindo ações de consulta, filtragem, criação de tabelas, entre outros, por meio de linha de comandos. É suportada por vários sistemas operativos. Possui uma versão *open source* que suporta diversos tipos de bases de dados relacionais. A versão *Enterprise* suporta ainda bases de dados NoSQL. Uma das principais funcionalidades é a execução de *queries* SQL, possibilitando a importação/exportação, formatação, consulta e filtragem de resultados.

Outra das funcionalidades é a geração de modelos de dados a partir de bases de dados existentes, reconhecendo as ligações entre tabelas e a definição de chaves primárias e chaves estrangeiras [81 – 83].

4.4.Grafana

O Grafana é uma plataforma web *open source* que permite a monitorização de sistemas através de análise e visualização interativa de dados. Muito popular na análise de dados em séries temporais, possui uma grande quantidade de plug-ins que permitem utilizar dados de diversas fontes e formatos, desde *logs, reports,* ficheiros CSV, entre outros.

O editor presente no Grafana fornece uma interface personalizada ao utilizador que auxilia na escrita de *queries* sob os dados que suportam a visualização interativa.

Adicionalmente, permite a prototipagem e desenvolvimento gráfico com recurso aos *templates* ou através de opções de personalização que a plataforma dispõe. Os *dashboards* criados podem ser reutilizados e partilhados com os *stakeholders*. A atribuição e gestão de diferentes níveis de permissões garante que cada colaborador e/ou departamento tem apenas acesso à informação que necessita.

O Grafana já inclui algumas métricas de análise, contudo podem ser importadas ou criadas métricas. Possui um sistema de alerta, que funciona através da definição regras sob as métricas utilizadas e do envio automático de notificações, permitindo uma monitorização contínua [84, 85].

4.4.1. Formação

Sendo o Grafana uma ferramenta indispensável para a realização do trabalho e alcançar os objetivos propostos no estágio, foi necessário aprofundar o conhecimento sobre esta ferramenta e o seu modo de funcionamento, através da visualização de tutoriais, leitura de documentação existente e exploração da ferramenta mencionada.

O estudo começou com a leitura documentação existente do Grafana [86], onde foi possível perceber o melhor modo de instalação do *software* dentro das opções disponíveis. Foram também visualizados 2 tutoriais em formato de vídeo [87, 88] que em conjunto com a documentação acima mencionada permitiram configurar adequadamente o Grafana e explorar algumas das suas funções básicas. Também foi utilizado um *website,* desenvolvido pela mesma empresa do Grafana, que simula a interface do Grafana, permitindo assim a interação entre o utilizador e o *software* sem necessidade de realizar qualquer instalação e/ou

configuração, podendo, por isso, ser utilizado na realização de testes e demonstrações ou usado para fins de aprendizagem [89].

5. Conceção da Solução

Este capítulo descreve todas as etapas do processo de desenvolvimento da solução e das tarefas a ele associadas. O processo de desenvolvimento foi definido de acordo com os objetivos propostos no plano de estágio, em conjunto com a empresa, onde se estipularam as principais tarefas e o seu encadeamento.

5.1. Arquitetura da solução

A arquitetura é uma estrutura fundamental que detalha todos os componentes em que a solução está assente, bem como a forma como estes componentes interagem, para alcançar os requisitos impostos. Na Figura 11 encontra-se representada a arquitetura da solução de visualização de dados a implementar, onde se identificam os componentes, os processos e as ferramentas associadas.

O primeiro componente da solução consiste nas fontes de dados que servem para alimentar o *dashboard* final. É neste componente que se inserem os relatórios gerados pelas ferramentas de segurança. Nesta etapa existe uma observação e compreensão inicial dos dados existentes nos relatórios, de forma a entender que tipo de dados se encontra presente.

O processo seguinte consiste no desenvolvimento de um *script* em P*ython* para realizar parte do processo ETL, correspondente à extração dos dados. Como *outputs* deste processo são criados ficheiros em formato CSV, contendo os dados extraídos.

Um segundo *script* é desenvolvido para a transformação e preparação dos dados. Durante este processo existe uma seleção dos atributos extraídos, de acordo com a sua relevância. São ainda criados outros atributos, inexistentes nos relatórios de segurança, considerados relevantes para o *dashboard*. É realizada reestruturação final, de modo que a estrutura dos dados corresponda à estrutura presente na base de dados relacional.

O segundo componente consiste na base de dados relacional, que serve como repositório para os dados extraídos. Esta base de dados é construída com recurso ao DBeaver e à MariaDB [90], um sistema de gestão de bases de dados. Após a sua criação e configuração, as tabelas são devidamente preenchidas.

Primeiramente é desenhado um modelo de dados adequado, que considere as relações entre as tabelas de dados. Esse modelo serve de base para a criação da base de dados

relacional. O DBeaver possibilita a população das tabelas, através da criação de *tasks* que permitem semi-automatizar o processo de carregamento de dados, etapa final do processo de ETL.

O último componente da solução é o Grafana, a ferramenta de visualização de dados. Este componente interage com a base de dados através de um *plug-in*, utilizando os dados para criar as diferentes representações visuais que constituem o *dashboard*.

Foram utilizados dados provenientes de 3 projetos da empresa para a realização do trabalho de estágio, cujos nomes foram alterados para manter a confidencialidade da informação. As etapas de desenvolvimento da solução são detalhadas nas diferentes secções deste capítulo.

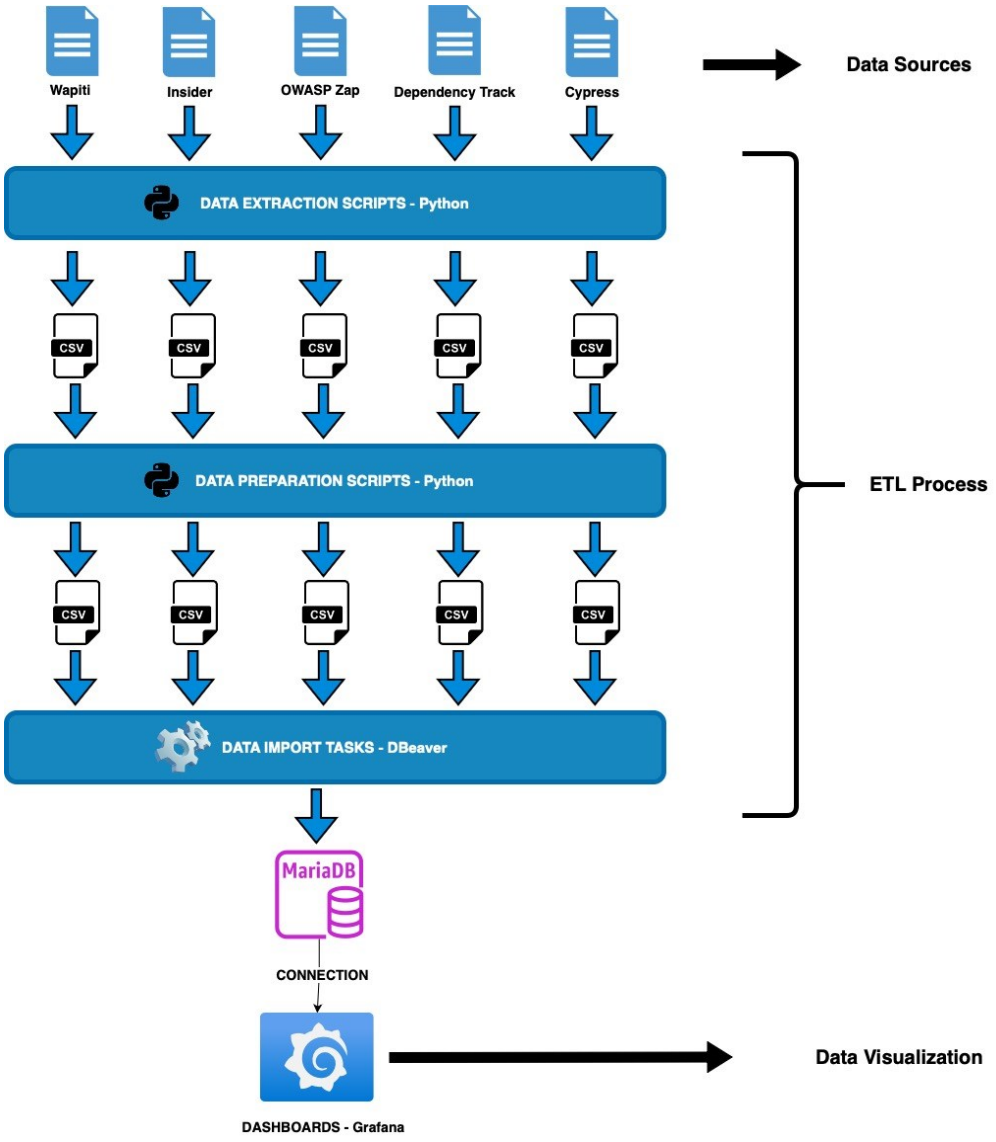


Figura 11 – Arquitetura da solução

5.2. Requisitos do sistema

Os requisitos referem-se às especificações que o sistema deve apresentar. Estes requisitos encontram-se divididos em dois grupos: requisitos funcionais, que descrevem as funcionalidades que a solução deve possibilitar aos seus utilizadores e os requisitos não funcionais, que descrevem as características que a solução apresenta de forma a poder operar nos vários ambientes. De seguida são descritos os requisitos da solução desenvolvida.

Requisitos Funcionais:

- Possuir uma vista geral de todas as vulnerabilidades do projeto.
- Possuir uma vista geral do número total de vulnerabilidades identificadas no projeto considerando um determinado período temporal.
- Pesquisar e filtrar vulnerabilidades de acordo com a ferramenta de segurança utilizada na sua identificação, pela sua classe de severidade e/ou de acordo com a data.
- Permitir aos utilizadores aceder aos detalhes das vulnerabilidades sempre que necessário (*details-on-demand*).
- Permitir a visualização da proporção e número de *features* e *scenarios* que passaram e falharam nos testes automáticos do Cypress, bem como ao número total de *features* e *scenarios* testados.
- Permitir o verificar o número de *features* testadas no Cypress ao longo do tempo.
- Visualizar a duração dos testes para cada *feature* e a duração total de todos testes do Cypress.

Requisitos Não Funcionais:

- É necessário ter acesso a *reports* das várias ferramentas de forma que o *script* possa extrair os seus dados.
- É necessário o uso da MariaDB como base de dados para guardar todos os dados relevantes.
- É necessário o uso do DBeaver para desenvolver os mecanismos de importação dos dados extraídos para a base de dados.
- É necessário o uso do Grafana para visualizar a informação através dos *dashboards* produzidos.
- A base de dados da solução deve ser capaz de lidar com dados de novas ferramentas de segurança e acomodá-los nas suas tabelas.
- A base de dados deve ser suficientemente flexível de forma a incorporar outros tipos de dados e atributos na sua estrutura.
- O acesso à informação disponibilizada nos *dashboards* deve ser controlado com base nos tipos de utilizadores, salvaguardando o acesso não autorizado aos dados.
- Toda a informação existente nos *dashboards* deve estar disponível a todos os utilizadores que possuam permissões de acesso adequadas.
- A solução deve ser intuitiva e fácil de utilizar por parte dos utilizadores com acesso autorizado.

5.3. Tipos de utilizadores

A solução desenvolvida foi projetada para ser utilizada principalmente por especialistas em segurança e engenheiros de QA. Propõe uma visão holística, permitindo que algumas vulnerabilidades descobertas sejam prontamente detetadas nas fases iniciais de desenvolvimento. Deste modo, através da análise dos resultados de uma ferramenta End-2-End (Cypress), a informação apresentada é mais precisa e contextualizada. Analistas de Segurança e Gerentes de Segurança da Informação utilizam a solução de forma a verificar e

analisar a informação de vulnerabilidades das aplicações. Os engenheiros de QA verificam e analisam a informação referente aos testes automáticos de integração do Cypress.

5.4. Observação e extração de dados

Antes de se proceder à manipulação e extração dos dados, foi necessário fazer uma observação *à priori* destes, através da leitura dos *reports* e *logs* gerados pelas ferramentas de segurança, de modo a haver uma compreensão sobre os dados extraídos e do seu significado.

A extração dos dados relevantes para as etapas seguintes do trabalho foi feita através do *Visual Studio Code* [91] , utilizando a linguagem *Python* [92], devido ao facto de possuir um vasto conjunto de bibliotecas adequadas à realização das operações necessárias deste processo [93].

Os scripts desenvolvidos para a extração de dados dos três projetos são idênticos, e algumas partes destes scripts não são relevantes para a explicação do processo (por exemplo, a importação de bibliotecas). Por essa razão, apenas são apresentados nesta secção, os excertos dos *scripts* de um dos projetos que contribuem para a explicação do processo. Os *scripts* completos podem ser consultados no Anexo D.

5.4.1. WSAP

Como referido anteriormente, o WSAP é uma combinação de 3 ferramentas de segurança, cada uma com a sua função. Depois de efetuados todos os testes das ferramentas é gerado um relatório global contendo os resultados de todos os testes efetuados, em formato JSON. Na Figura 12 pode ser visualizado uma parte do relatório criado pelo WSAP, correspondente aos dados de uma vulnerabilidade encontrada pela ferramenta Wapiti. Na secção 5.5 é feita uma descrição mais detalhada do significado dos dados obtidos.

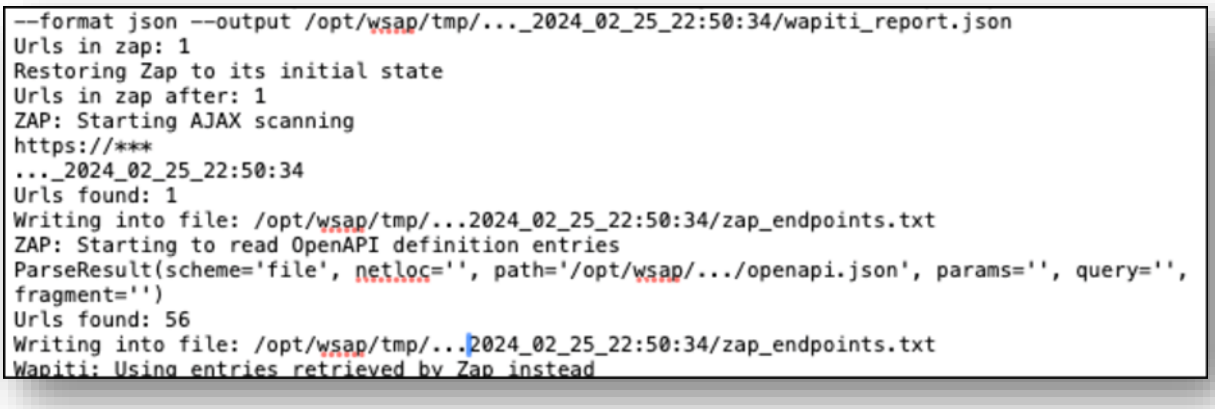


Figura 12 – Parte do *log* gerado pelo WSAP

O ficheiro do relatório foi descarregado manualmente da pasta da empresa para permitir o acesso e processamento do seu conteúdo. Optou-se por separar os dados por ferramenta, para trabalhar com um menor número de dados, pertencentes ao mesmo conjunto, e facilitar a sua manipulação.

O primeiro *script* desenvolvido permitiu separar os dados das três ferramentas, gerando três ficheiros de texto, cada um contendo os dados correspondentes de uma ferramenta.

Importou-se a biblioteca json para carregar e ler os dados do ficheiro, como é possível visualizar na Listagem 1.

```
1. import json
2.
3. projeto_nome="/Users/alexandre.penela/Documents/Report
Extration/WSAP/Project_A/Project_A.json"
4. f=open(projeto_nome)
5. data=json.load(f)
```

Listagem 1 – Importação de bibliotecas e carregamento do ficheiro do relatório do WSAP

Foram criados 3 dicionários, representados na Listagem 2, cada um correspondente aos dados de cada uma das ferramentas.

```
1. wapiti=data["DAST_WAPITI"]
2. zap=data["DAST_ZAP"]
3. insider=data["SAST_InsiderCLI"]
```

Listagem 2 – Dicionários de dados das ferramentas do WSAP

No passo seguinte, representado na Listagem 3, os dicionários foram transformados em objetos JSON, permitindo assim serem escritos num ficheiro deste formato.

```
1. json_wapiti=json.dumps(wapiti)
2. json_zap=json.dumps(zap)
3. json_insider=json.dumps(insider)
```

Listagem 3 – Transformação dos dicionários de dados em formato JSON *object*

No final os dados de cada uma das ferramentas foram exportados para diferentes ficheiros, para que a sua extração e processamento seja efetuado de forma independente (Listagem 4).

```
1. with open( r"/Users/alexandre.penela/Documents/Report
Extration/WSAP/Project_A/wapiti_project_A.json", "w") as dados_wapiti:
2.     dados_wapiti.write(json_wapiti)
3.
4. with open( r"/Users/alexandre.penela/Documents/Report
Extration/WSAP/Project_A/zap_project_A.json", "w") as dados_zap:
5.     dados_zap.write(json_zap)
6.
7. with open( r"/Users/alexandre.penela/Documents/Report
Extration/WSAP/Project_A/insider_project_A.json", "w") as dados_insider:
8.     dados_insider.write(json_insider)
```

Listagem 4 – Exportação de dados das ferramentas do WSAP

Foram desenvolvidos 3 *scripts* de extração, 1 para cada ferramenta, que apresentam grandes semelhanças entre si, distinguindo-se essencialmente no número e nome de variáveis que são extraídas. Assim, é apresentado apenas um exemplo do *script* para uma das ferramentas.

Foram importadas as bibliotecas necessárias e carregado o ficheiro com os dados de uma das ferramentas. Em seguida, como apresentado na Listagem 5, foram criadas listas vazias para cada um dos atributos existentes no relatório, onde a informação extraída é adicionada.

```
1. Vulnerabidade = []
2. Vulnerability_type = []
3. Info = []
4. curl_command = []
5. http_request = []
6. level = []
7. method = []
8. wstg = []
```

Listagem 5 – Criação de listas para os dados de cada atributo

Na Listagem 6 encontra-se representado o *loop* de extração de dados criado, que percorre toda a informação do ficheiro e anexa os dados das variáveis à lista correspondente.

```
1. for key in data:
2.     for vul in data[key]:
3.         Vulnerabidade.append(key)
4.         curl=vul.get('curl_command', None)
5.         http=vul.get('http_request', None)
6.         inf=vul.get('info', None)
7.         vul_type=vul.get('vulnerability_type', None)
8.         lev=vul.get('level', None)
9.         met=vul.get('method', None)
10.         wst=vul.get('wstg', None)
11.         curl_command.append(curl)
12.         http_request.append(http)
13.         Info.append(inf)
14.         Vulnerability_type.append(vul_type)
15.         level.append(lev)
16.         method.append(met)
17.         wstg.append(wst)
```

Listagem 6 – Loop de extração de dados da ferramenta Wapiti

Em seguida, como pode ser observado na Listagem 7, foi criado um *dataframe*, utilizando como colunas todas as listas que contêm a informação extraída. Esse *dataframe* final foi exportado para um ficheiro em formato CSV.

```
1. dwap= {'VULNERABILITY_CLASS': Vulnerabidade,'VULNERABILITY_TYPE':Vulnerability_type,
'INFO':Info,  'CURL COMMAND TO REPLICATE': curl_command, 'HTTP_REQUEST': http_request, '
SEVERITY_LEVEL':level, 'METHOD':method, 'WSTG':wstg}
2.
3. df=pd.DataFrame(dwap)
4. df=df.replace('\n', '',regex=True)
5.
6. ficheiro_csv="/Users/alexandre.penela/Documents/Report
Extration/WSAP/Project_A/wapiti_extract_Project_A.csv"
7. df.to_csv(ficheiro_csv, index=False)
```

Listagem 7 – Exportação dos dados extraídos do Wapiti

5.4.2. Cypress

Os dados do relatório gerado pelo Cypress foram extraídos através de três *scripts*, extraindo separadamente a informação relativa às *features* testadas na aplicação, a informação referente aos cenários testados e os dados referentes aos passos dentro de cada cenário.

No *script* utilizado na extração das *features* importaram-se as bibliotecas necessárias e carregou-se o ficheiro do *report*. Foi criado um conjunto de listas vazias para armazenar os dados de cada campo presente no relatório.

Na Listagem 8 é possível visualizar o processo de extração dos dados, realizada por meio de duas funções, uma especificamente dedicada à extração da duração total dos testes realizados em cada *feature* e outra responsável por extrair a informação dos atributos.

```
Nome_feat=[]
desc_feat=[]
falhas=[]
aprovados=[]
estado=[]
Duracao=[]

for i in dados :
    i= str(i)
    des=i.find("'description':") #descrição
    des_fin=i.find("elements")
    desc=i[des+20:des_fin-4]
    descript=desc[0:1023]
    desc_feat.append(descript)
    id=i.find("'id'")   #Nome
    id_fin=i.find(';',id)
    feat=i[id+5:id_fin].replace('-',' ').replace("'","")
    Nome_feat.append(feat)

    volta=i.count("Scenario") #Cenários aprovados e falhados
    pa=int(volta)
    scene=i.find("Scenario")
    scene_fin=i.find("Scenario",scene+1)
    fail=i.count("failed",scene,scene_fin)
    if fail > 0:
        pa=pa-1
    else:
        pa=pa
    volta= volta -1
    while volta > 0:
        scene=i.find("Scenario",scene_fin)
        scene_fin=i.find("Scenario",scene+1)
        fail=i.count("failed",scene,scene_fin)
        if fail > 0:
            pa=pa-1
        else:
            pa=pa
        volta= volta -1
    aprovados.append(pa)
    falha=int(i.count("Scenario"))-int(pa)
    falhas.append(falha)
    if falha > 0: #Coluna Estado
        est = "Failed"
    else:
        est= "Passed"
    estado.append(est)
```

```
47.
48. def soma_duracao(dici):
49.     total=0
50.     for elemento in dici['elements']:
51.         for passo in elemento['steps']:
52.             total += passo['result']['duration']
53.     return total
54.
```

Listagem 8 – Função de extração de dados das *features* do Cypress

Os dados da duração foram transformados de forma a se obter a duração em segundos. Este processo de transformação encontra-se representado na Listagem 9.

```
1. for i in dados:
2.     soma_duracao(i)
3.     soma_total=soma_duracao(i)/1000000000
4.     Duracao.append(round(soma_total))
```

Listagem 9 – Transformação da duração do tempo de testes em segundos

Após a anexação dos dados extraídos às respetivas listas, estas são utilizadas como colunas na constituição de um *dataframe*. A este *dataframe* é adicionada uma coluna com a data do relatório, em formato dd/mm/aaaa. O *dataframe* final é exportado para um ficheiro em formato CSV. Este processo está representado na Listagem 10.

```
 1. #Tabela
 2. cypress_feat= {'Name': Nome_feat, 'Description': desc_feat, 'Status': estado,
'scenario_aproved':aprovados, 'Scenario_failed':falhas, 'Duration (s)':Duracao}
 3.
 4. df=pd.DataFrame(cypress_feat)
 5.
 6. #Variável data
 7.
 8. data= date.today()
 9. df['Date']= data
10. df["Date"]= pd.to_datetime(df["Date"], dayfirst= True)
11. df["Date"]=df["Date"]
12.
13. #Exportar output para csv
14. ficheiro_csv="/Users/alexandre.penela/Documents/Report
Extration/cypress/Project_A/features/tabela_features.csv"
15. df.to_csv(ficheiro_csv, index=False)
```

Listagem 10 – Criação do *dataframe* e exportação dos dados das *features* do Cypress

A Listagem 11 apresenta o *script* desenvolvido para a extração da informação de cada cenário. Este *script* é semelhante ao anterior, em termos de código, diferenciando-se apenas pela criação uma nova variável, designada de "*Feature_ID*", que associa o cenário extraído à *feature* à qual pertence.

```
feature_id=[]
cenario=[]
step_aproved= []
step_failed = []
estado=[]
tempo=[]

feature=1

def dados_extrat(dados):

    for elemento in dados['elements']:
        cen=elemento['name']
        cenario.append(cen)
        passed=0
        failed=0
        dur=0
        for passo in elemento['steps']:
            dur += passo['result']['duration']
            if passo['result']['status'] == "passed":
                passed +=1
            else:
                failed += 1
        if failed > 0 :
            est= "failed"
        else:
            est= "passed"
        total=int(dur)/1000000000
        tempo.append(round(total,ndigits=1))
        step_aproved.append(passed)
        step_failed.append(failed)
        estado.append(est)
        feature_id.append(feature)

#Extração de dados dos cenários de cada feature

for i in dados:
    dados_extrat(i)
    feature=feature +1
```

Listagem 11 – Função de extração de dados dos cenarios de cada *feature*

O *script* de extração da informação relativa aos passos apenas difere na criação da variável criada ser relativa ao cenário "*Scenario_ID*" em vez de ser relativa à *feature*. Optou-se, desta forma, por não apresentar este *script* nesta secção e adicioná-lo ao Anexo D.

5.4.3. Dependency Track

O processo de extração de dados do Dependency Track segue uma abordagem semelhante às anteriores, onde são importadas as bibliotecas, é efetuado o carregamento do ficheiro e criado um conjunto de listas para guardar os dados extraídos.

Na criação da função de extração de dados, representada na Listagem 12, foi necessário considerar um pormenor importante: a existência de dois mecanismos de classificação de vulnerabilidades, o CVSS2 e o CVSS3, sendo este último o mais recente.

As vulnerabilidades identificadas há mais tempo foram classificadas segundo o CVSS2. Enquanto algumas já foram reclassificadas de acordo com o novo método, possuindo assim a informação de ambos os mecanismos, outras apenas têm a informação relativa primeiro método. Por outro lado, vulnerabilidades mais recentes foram classificadas exclusivamente pelo CVSS3 ou ainda não possuem qualquer tipo de classificação.

Tendo em conta que uma vulnerabilidade pode ter duas pontuações e classes de severidade distintas, correspondentes aos dois métodos, a função de extração dá prioridade à informação do CVSS3, uma vez que é o método atualmente em vigor. Durante a extração de dados é verificado se existe informação relativa ao CVSS3. Caso a vulnerabilidade apresente essa informação, ela é extraída. Caso não hajam dados referentes aos CVSS3, é analisada a presença dos dados CVSS2. Caso não exista informação de nenhum dos métodos, é atribuído um *None value* à lista.

```
ref=[]
id=[]
source_name=[]
souce_url=[]
score_CVSS=[]
severity_CVSS=[]
method=[]
vector=[]
description=[]

for vul in dados ['vulnerabilities']:
    ref.append(vul['bom-ref'])
    cve=vul['id']
    id.append(cve)
    source_name.append(vul['source']['name'])
    url_= vul['source']['url']
    url_source=url_ + 'vuln/detail/' + str(cve)
    souce_url.append(url_source)
    des=vul['description']
    description.append(des[0:1023])
    has_cvssv3 = False
    has_cvssv2 = False
    for rating in vul ['ratings']:
        if rating['method'] == 'CVSSv3':
            has_cvssv3 = True
            score_CVSS.append(rating['score'])
```

```
27.             severity_CVSS.append(rating['severity'])
28.             method.append(rating['method'])
29.             vector.append(rating['vector'])
30.             break
31.
32.     if not has_cvssv3:
33.         for rating in vul['ratings']:
34.             if rating['method'] == 'CVSSv2':
35.                 has_cvssv2 = True
36.                 score_CVSS.append(rating['score'])
37.                 severity_CVSS.append(rating['severity'])
38.                 method.append(rating['method'])
39.                 vector.append(rating['vector'])
40.                 break
41.
42.             else:
43.                 score_CVSS.append(None)
44.                 severity_CVSS.append("NA")
45.                 method.append("NA")
46.                 vector.append("NA")
```

Listagem 12 – Função de extração de dados do Dependency Track

Posteriormente, é criado um *dataframe* com os dados extraídos, ao qual se adiciona a informação do tipo de componente testado pela ferramenta. Por fim o *dataframe* é exportado para um ficheiro em formato CSV.

5.5.Seleção de atributos

Dos dados extraídos, foram selecionados os atributos da tabela considerados relevantes para o *dashboard*. Esta seleção foi realizada em conjunto com o chefe do departamento de Segurança de Informação, tendo em conta as necessidades da empresa. Os restantes atributos foram removidos manualmente dos ficheiros CSV. A Tabela 1 contém os atributos finais selecionados do Wapiti, uma pequena descrição de cada um, um exemplo e o tipo de dados. As tabelas com os atributos finais das restantes ferramentas encontram-se no Anexo E.

Tabela 1 – Atributos finais da ferramenta Wapiti

ATRIBUTOS	DESCRIÇÃO	EXEMPLO	TIPO DE DADOS
VULNERABILITY_CLASS	Classe da vulnerabilidade	Low	Varchar

VULNERABILITY_TYPE	Tipo de vulnerabilidade encontrada	Content Security Policy Configuration	Varchar
INFO	Informação sobre a vulnerabilidade	CSP is not set	Varchar
HTTP_REQUEST	Pedido solicitado ao servidor	GET / HTTP/1.1 host: test-xxx.xxx.xxx\nconnection: keep-alive\nuser-agent: Mozilla/5.0 (Windows NT 6.1; rv:45.0) Gecko/20100101 Firefox/45.0 accept-language: en-US accept-encoding: gzip deflate br accept: text/htmlapplication/xhtml+xmlapplication/xml;q=0.9image/webp*/*;q=0.8	Varchar
SEVERITY_LEVEL	Nível da vulnerabilidade	7.4	Decimal
WSTG	Código WSTG associado ao tipo de vulnerabilidade Código WSTG associado ao tipo de vulnerabilidade	[WSTG-CONF-12 OSHP-Content-Security-Policy]]	Varchar

5.6. Transformação de dados

Os dados extraídos das ferramentas Wapiti, Insider, OWASP Zap e Dependency Track sofreram um processo de transformação e reestruturação, através de um segundo *script*. Este *script* adapta a estrutura dos dados, que se encontra num formato tabular, para um formato relacional, fazendo a correspondência com a estrutura da base de dados relacional (ver

secção seguinte). Esta transformação é essencial, pois facilita posteriormente o carregamento da informação na base de dados.

Uma vez que o *script* de transformação de dados é idêntico para todas as ferramentas, é apresentado apenas um exemplo deste *script* na Listagem 13. São importadas as bibliotecas necessárias e procede-se ao carregamento do ficheiro contendo os dados extraídos. São criadas listas para armazenar os dados na nova estrutura. Um *loop* precorre os dados do ficheiro e adiciona-os às listas correspondentes. Por fim, é criado um *dataframe*, com uma estrutura semelhante à base de dados relacional. É adicionada uma coluna com a data do relatório, no formato dd/mm/aaaa. Os dados transformados são exportados para um ficheiro CSV.

```
 1. input= pd.read_csv("/Users/alexandre.penela/Documents/Report
Extration/WSAP/Project_A/Ficheiros extraidos/wapiti_extract_Project_A.csv", sep=";")
 2. input= input.replace('\n','', regex=True)
 3.
 4. vul=[]
 5. vul_id = []
 6. atr_id = []
 7.
 8. vul_n=1
 9. for i in range (0,len(input)):
10.     global vul_n
11.     dados = input.values[i]
12.     atribute = 1
13.     for valores in range(0, len(dados)):
14.         vul.append(dados[valores])
15.         vul_id.append (vul_n)
16.         atr_id.append(atribute)
17.         atribute += 1
18.     vul_n +=1
19.
20. vulnerabilidades = {'Vulnerability_ID':vul_id, 'Atribute_ID':atr_id, 'Value': vul }
21. df=pd.DataFrame(vulnerabilidades)
22.
23. data= date.today()
24. df['Date']= data
25. df["Date"]= pd.to_datetime(df["Date"], dayfirst= True)
26. df["Date"]=df["Date"]
27.
28. df=df.dropna()
29.
30. df.to_csv('/Users/alexandre.penela/Documents/Report
Extration/WSAP/Project_A/dados_A_wapiti.csv', index=False)
```

Listagem 13 – *Script* de transformação de dados

5.7.Desenho do modelo de dados

A ferramenta online draw.io [94] foi utilizada para desenvolver o modelo de dados relacional a ser implementado. Este modelo passou por várias versões até se atingir um modelo estável.

Por simplicidade, apenas é apresentada a primeira e a versão otimizada do modelo, na Figura 13 e Figura 14, respetivamente.

O 1º modelo é constituído por 7 tabelas, 1 para cada ferramenta, com exceção do Cypress, que apresenta 2 tabelas, uma relativa às *features*, outra relativa aos cenários e aos *steps* e ainda uma tabela relativa aos projetos, contendo um "*Project_ID*" e o "*Project_name*".

Após uma primeira análise do modelo, verificou-se que este não tinha escalabilidade suficiente no sentido de integrar dados de novas ferramentas de segurança, caso haja a necessidade futuramente e seria, por isso, necessário criar novas tabelas para adicionar os dados dessas novas ferramentas. Com base nesse critério foram desenvolvidos novos modelos até à construção do modelo final.

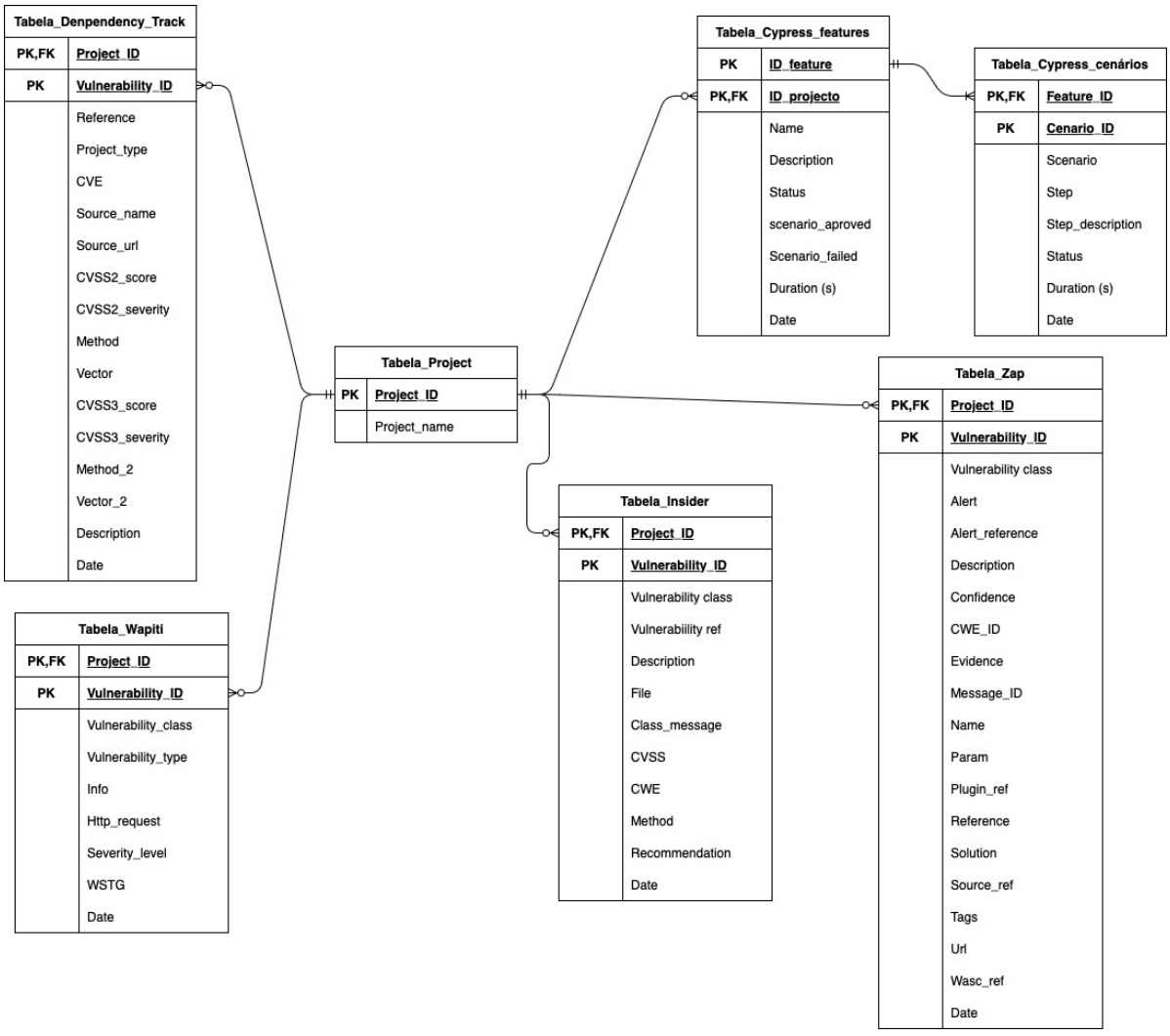


Figura 13 – Modelo relacional de dados inicial

O modelo final, representado na Figura 14, garante a escalabilidade pretendida, permitindo assim a integração de novas ferramentas e novos atributos na base de dados, sem a necessidade de incrementar o número de tabelas.

Este modelo é composto por 8 tabelas. No lugar de tabelas para cada ferramenta, existem 4 tabelas diferentes, uma tabela relativa às ferramentas, uma relativa aos atributos extraídos, uma para as vulnerabilidades e uma contendo os valores dos atributos. O número de tabelas referentes ao Cypress aumentou, estando agora a informação distribuída em 3 tabelas distintas, em vez das 2 iniciais. Esta alteração diminui tempo de processamento da *query* realizada, uma vez que o número de operações necessárias sob os dados a serem visualizados diminui (*count*, *sum*, etc). Existe, deste modo, uma tabela referente às *features*, outra aos cenários e a última com os dados de cada *step*.

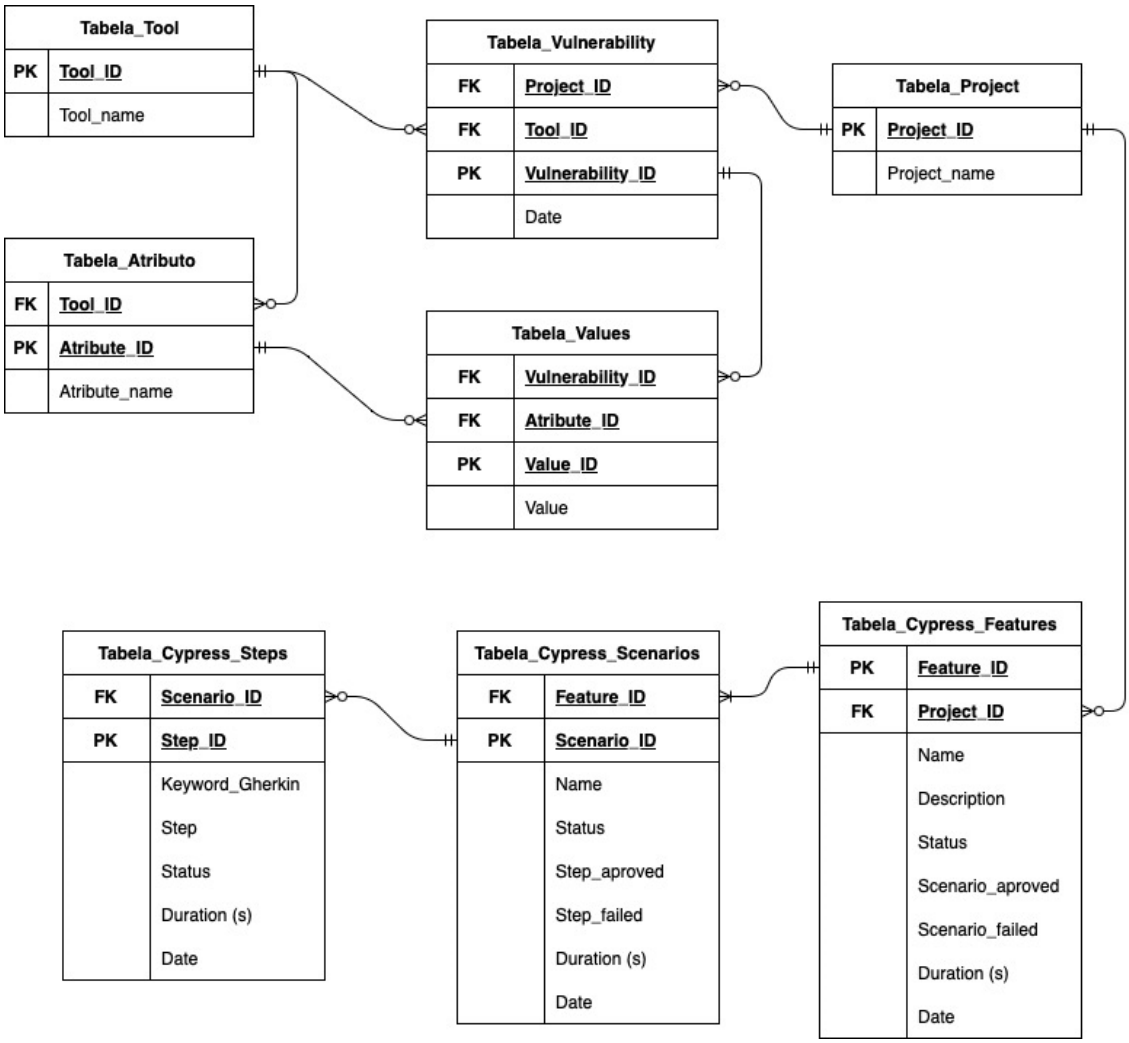


Figura 14 – Modelo relacional de dados final

5.8.Criação da Base de Dados e importação dos dados

As tabelas identificadas no modelo relacional foram geradas com recurso ao DBeaver, que é capaz de interagir com o servidor de gestão de bases de dados MariaDB utilizado pela empresa. Para cada atributo das tabelas foi definido o tipo de dados, se permitia a ausência de valores (nulos) e se o valor era sequencial. Foram também definidas a chave primária e estrangeira/s para cada uma das tabelas. Na Figura 15 encontra-se representado, como exemplo, as propriedades da tabela *Vulnerability* e dos seus atributos. As propriedades das restantes tabelas da base de dados podem ser visualizadas no Anexo F.

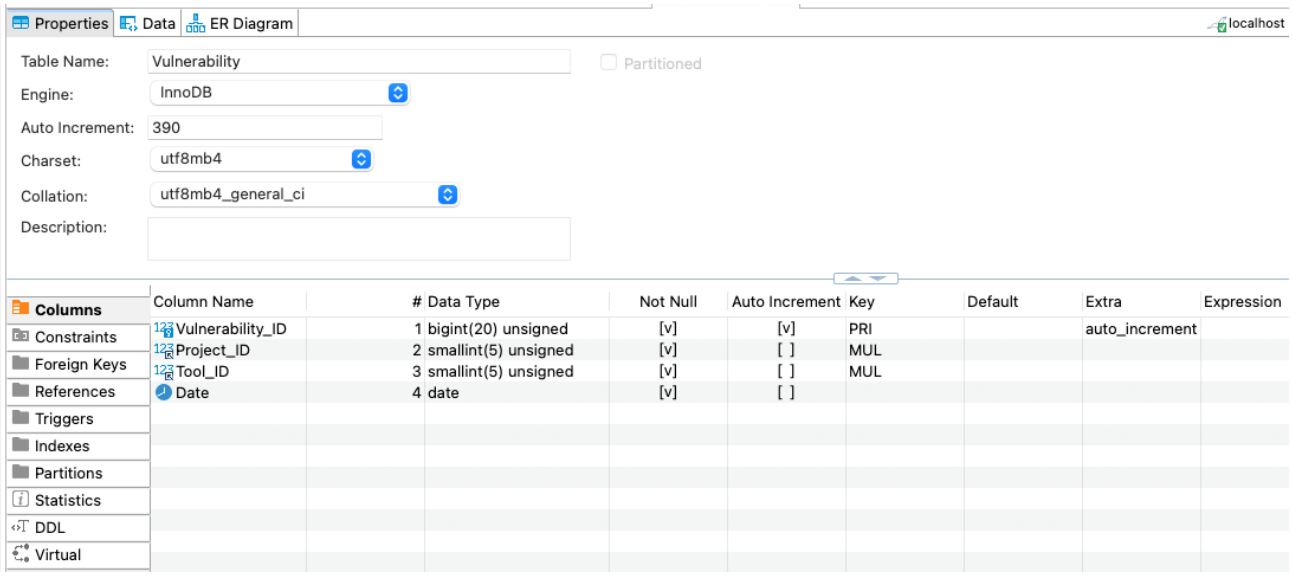


Figura 15 – Propriedades da tabela *Vulnerability*

Estabeleceu-se o relacionamento entre as tabelas através das chaves estrangeiras. Uma das funcionalidades do DBeaver é a criação de *tasks* que permitem a semi-automatização de tarefas repetitivas como é o caso da importação de dados. No painel de configuração é definido o tipo de tarefa que se pretende realizar.

Na importação de dados é selecionado o ficheiro que servirá de fonte de dados e a tabela alvo da base de dados. Existe ainda um conjunto de propriedades do ficheiro a importar que podem ser fornecidos pelo utilizador, conforme demonstrando na Figura 16. Neste painel a data foi definida para o formato "dd/MM/yyyy", correspondendo-o ao formato que se encontra no ficheiro de importação.

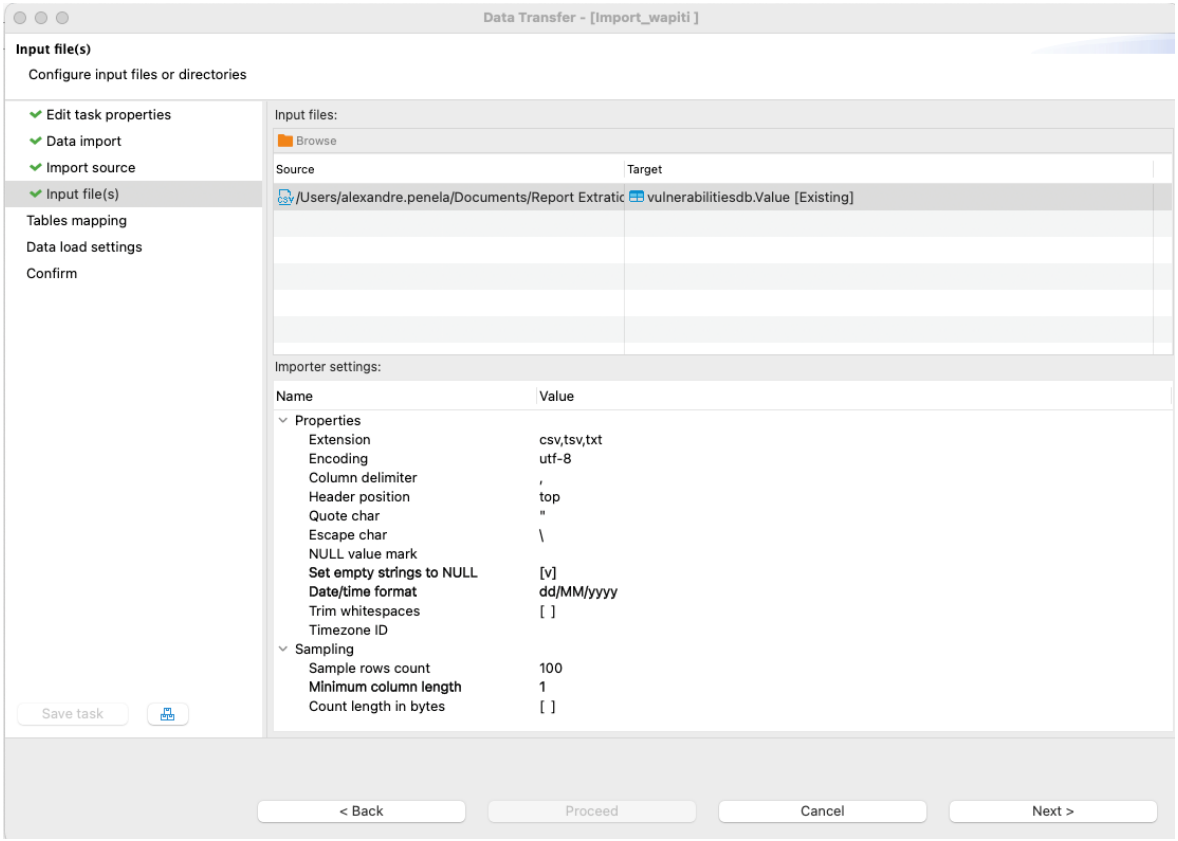


Figura 16 – Painel de configuração do ficheiro de importação

No painel seguinte, ilustrado na Figura 17, é feito um *mapping*, onde se seleciona e corresponde os campos da tabela de origem com os campos da tabela da base de dados. Existe a opção de *preview* que permite observar o resultado da importação dos dados na tabela alvo. No final aparece um resumo dos parâmetros da *task*. Confirmando a criação da tarefa, esta fica numa lista de tarefas configuradas, bastando um clique para que a *task* seja executada.

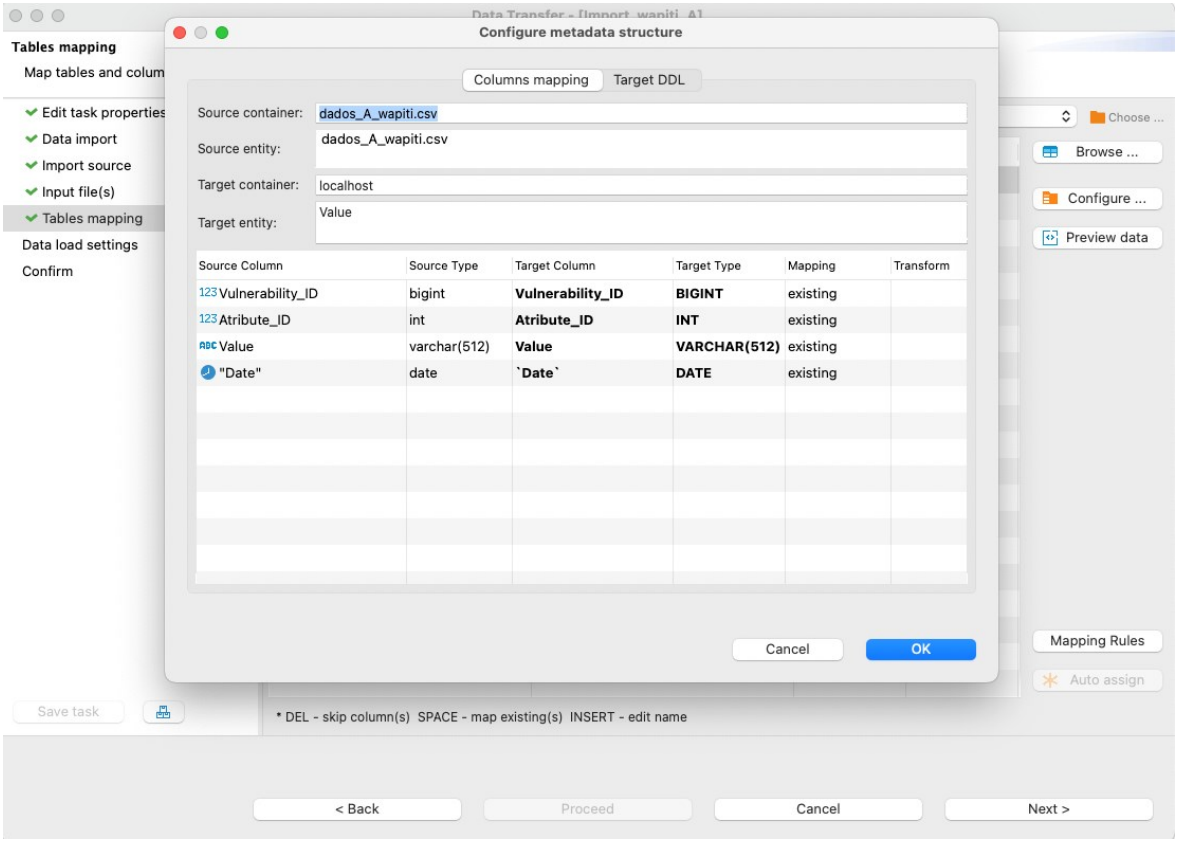


Figura 17 – Painel de mapeamento dos dados

5.9. Visualização de dados

O Grafana, ferramenta de visualização utilizada, foi selecionada tendo em consideração alguns critérios definidos pela empresa e devidamente descritos nos requisitos (ver secção 5.2).

Inicialmente, estabeleceu-se uma ligação do Grafana para com a base de dados MySQL usando um *plug-in* dedicado. Foi definida a estrutura e a informação presente em cada *dashboard*, com base nos objetivos da empresa. O Grafana permite a utilização de 2 temas de visualização pré-definidos, o modo escuro (*Dark theme*) e o modo mais claro (*Light theme*), existindo ainda possibilidade de criar e implementar temas mais personalizados. O modo D*ark* e o modo L*ight,* representados na Figura 18 e Figura 19, respetivamente.

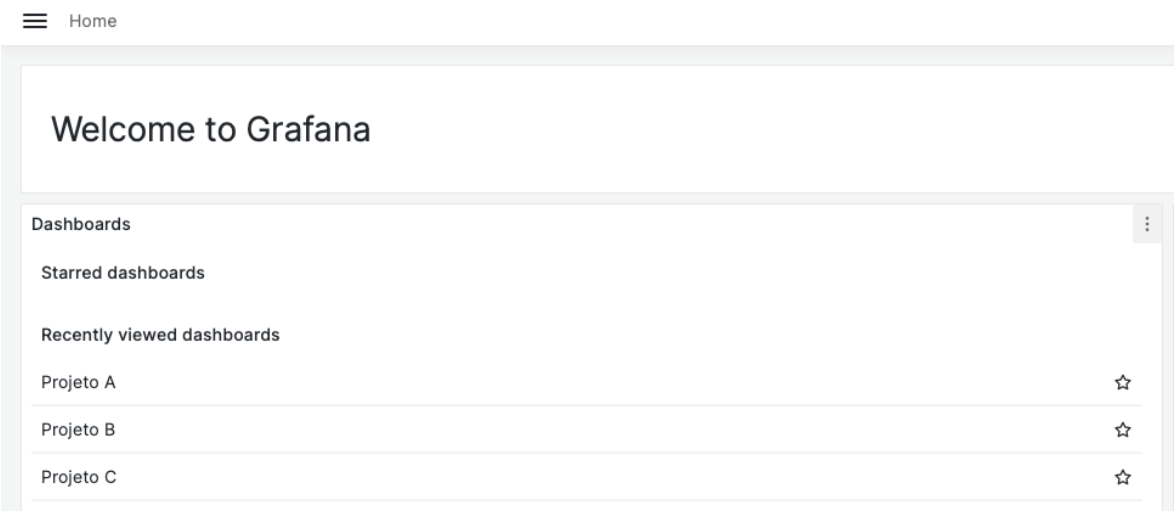


Figura 18 – Layout principal do Grafana em modo *light*

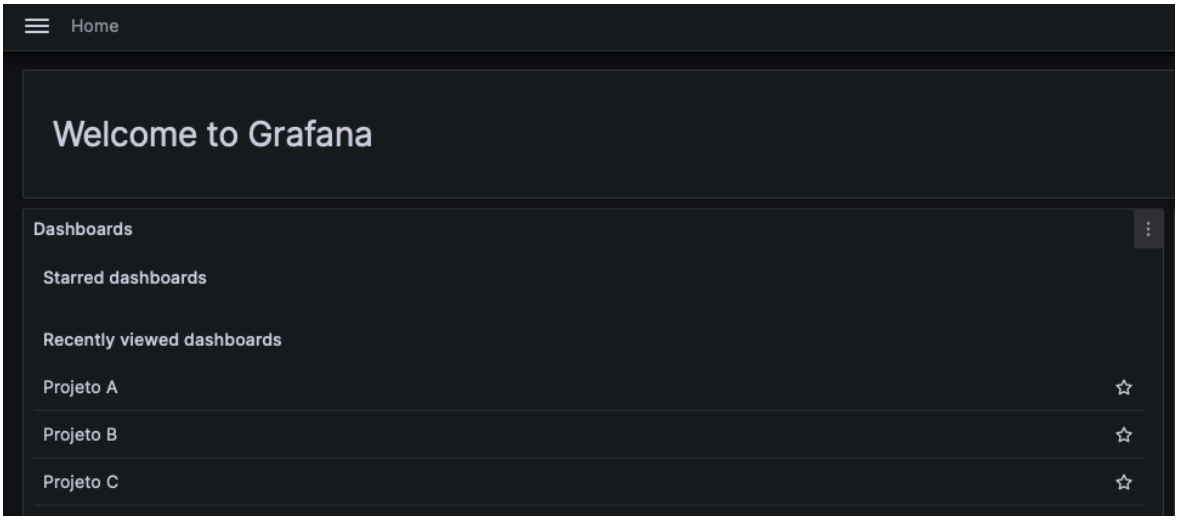


Figura 19 – Layout principal do Grafana em modo *dark*

Para efeitos de explicação do *dashboard* e dos vários painéis que o compõem será utilizado o modo L*ight*. Os *dashboards* apresentam a informação de cada um dos projetos, dividido em 3 secções principais, que estão divididas em separadores, como pode ser observado na Figura 20.

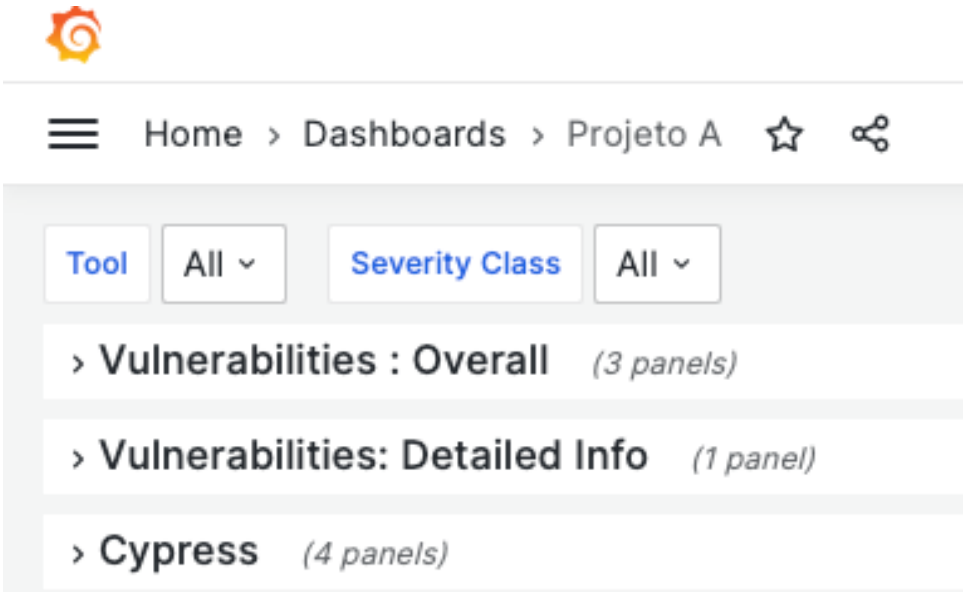


Figura 20 – Secções que compõem o *dashboard* do projeto A

A Figura 21 ilustra o *dashboard* completo e todas as secções que o compõem. A primeira secção do *dashboard*, apresenta informação mais generalizada das vulnerabilidades encontradas pelos *scanners* de segurança no projeto. A segunda secção mostra a mesma informação de uma forma mais detalhada. A última secção apresenta a informação relativa ao Cypress. De seguida será explicada mais detalhadamente cada uma das secções.

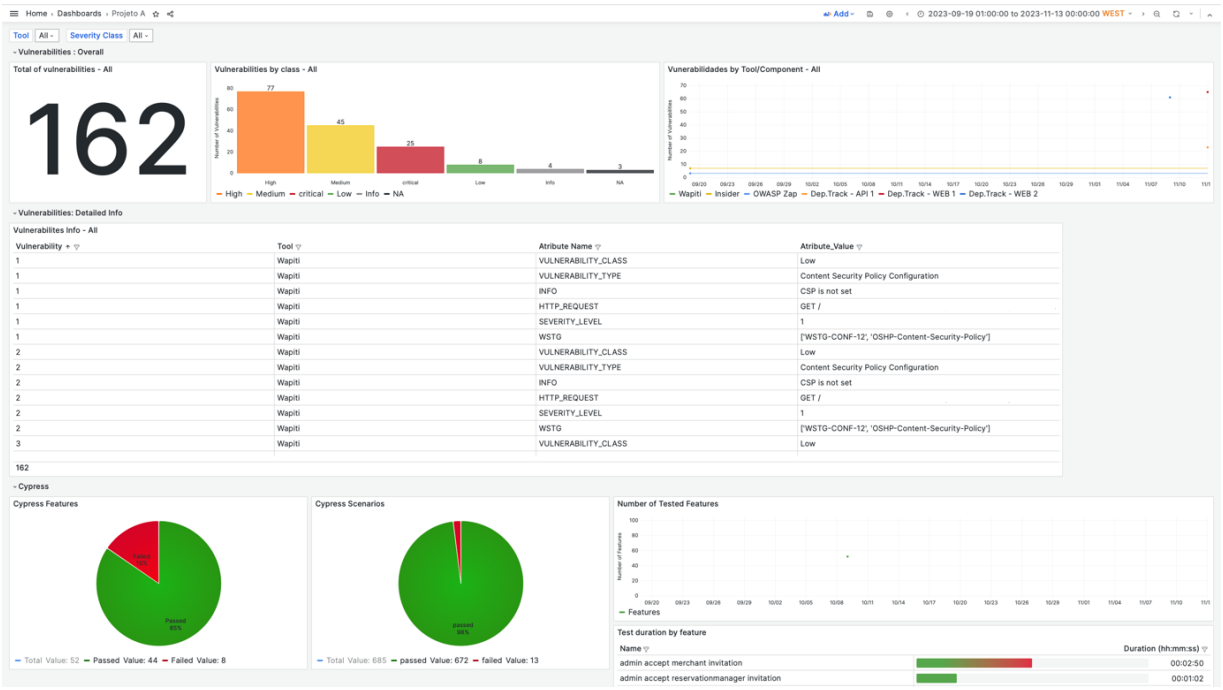


Figura 21 – *Dashboard* completo de segurança do projeto A

5.9.1. Vulnerabilities: Overall

A primeira secção do *dashboard*, composta por 3 painéis de visualização. A Figura 22 apresenta a primeira visualização mais à esquerda do *dashboard*, um cartão de valor único com a informação do número total atual de vulnerabilidades encontradas o projeto.

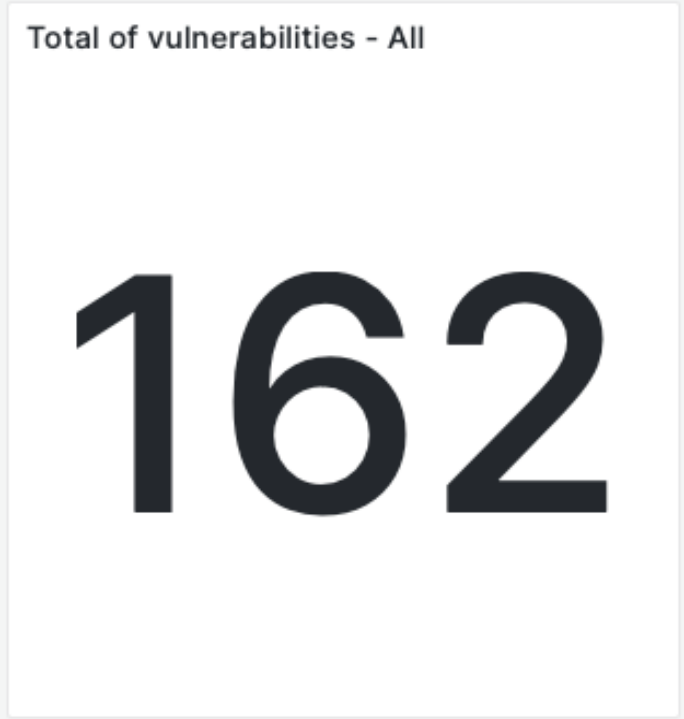


Figura 22 – Cartão de valor único com o total de vulnerabilidades do projeto

Na zona central da primeira secção encontra-se um gráfico de barras, representado na Figura 23 que divide as vulnerabilidades encontradas nas várias classes de severidade a que estas pertencem. Estas classes estão diretamente relacionadas com o seu CVSS. Os valores estão organizados por ordem decrescente, com a classe que apresenta o maior número de vulnerabilidades mais posicionada à esquerda e a classe com menor número disposta mais à direita.

As cores das barras estão, intencionalmente, associadas à classe de severidade que representam: classes de severidade mais baixa são exibidas com barras de cor verde ou cinza, enquanto as barras alteram a sua cor de forma gradual para tons de vermelho à medida que a severidade aumenta. Esta configuração permite ao utilizador ler, analisar e interpretar de forma fácil a informação, permitindo identificar a classe com maior número de

vulnerabilidades ou identificar o número de vulnerabilidades de severidade critica, através da associação vermelho a perigo por parte do ser humano [95].

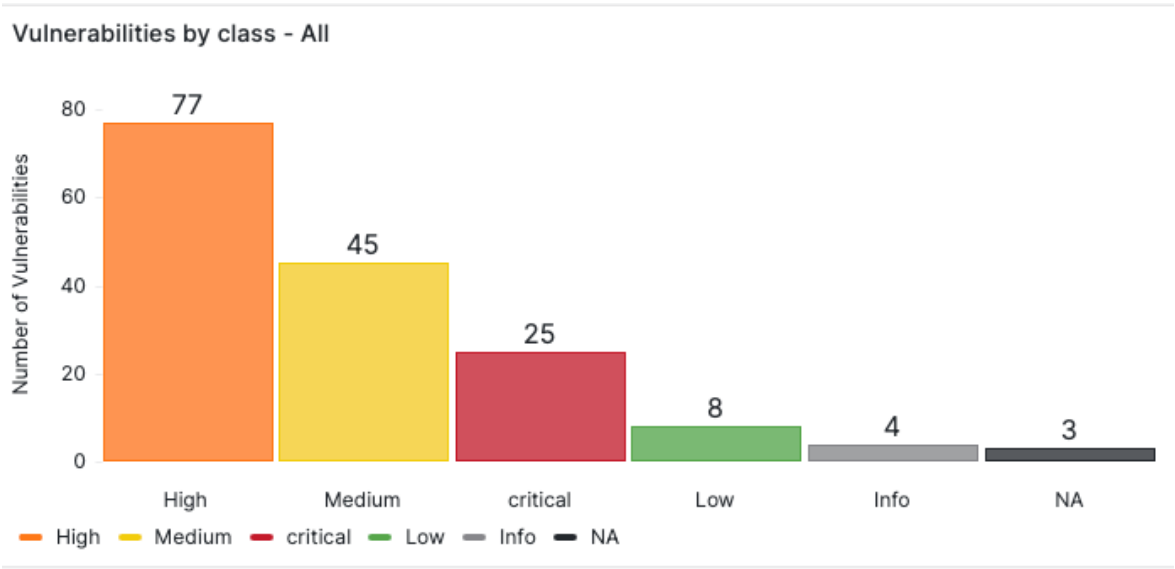


Figura 23 – Gráfico de barras do número de vulnerabilidades por categoria de severidade

O último painel, representado na Figura 24, corresponde a um gráfico de linhas que separa as várias vulnerabilidades do projeto pelo *scanner* de segurança onde foram identificadas, ao longo de um período temporal, representado no eixo das coordenadas. Este painel introduz a dimensão temporal no *dashboard*, uma vez permite observar a variação do número de vulnerabilidades identificadas no projeto por cada ferramenta de segurança ao longo do tempo.

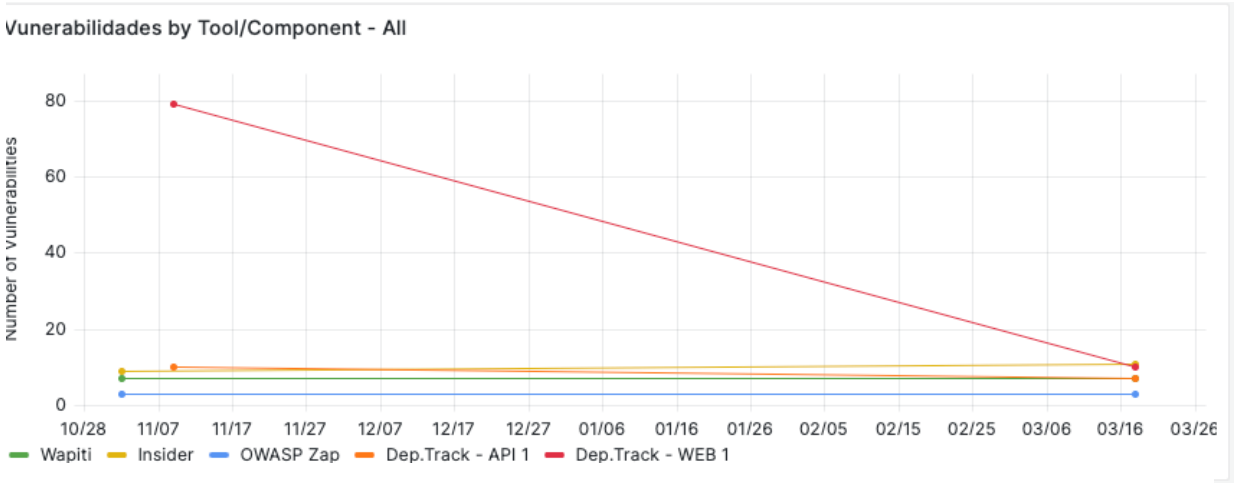
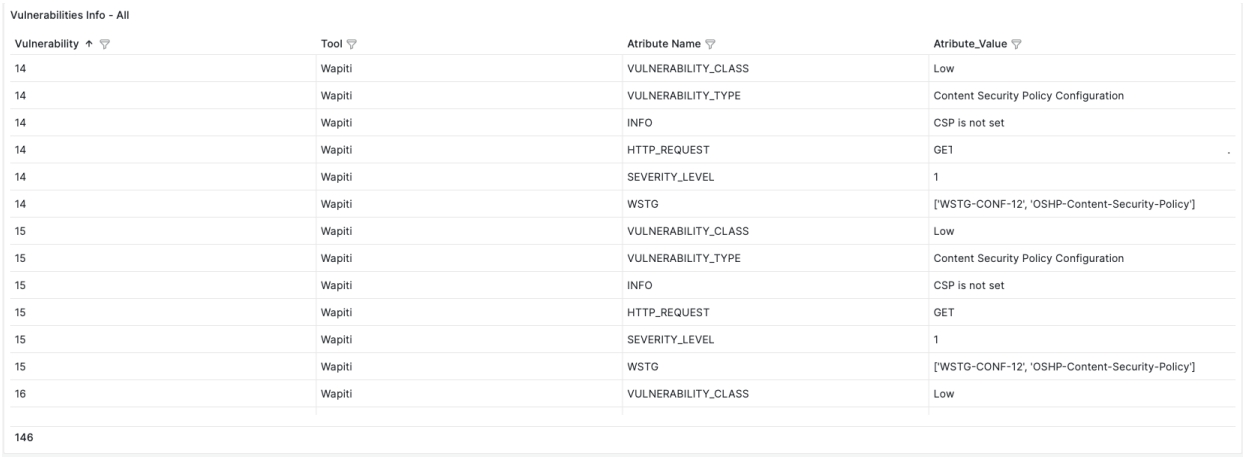


Figura 24 – Gráfico de linhas do número de vulnerabilidades por *scanner de* segurança ao longo do tempo

5.9.2. Vulnerabilities: Detailed Info

A segunda secção do *dashboard* contém uma tabela com a informação mais detalhada de cada uma das vulnerabilidades. A tabela contém filtros incorporados de forma a filtrar a informação e visualizar apenas a informação especifica que o utilizar pretende. A tabela encontra-se representada na Figura 25. Esta secção permite explorar e analisar a informação relativa às vulnerabilidades de uma forma mais aprofundada.



Vulnerabilities Info - All

Vulnerability	Tool	Atribute Name	Atribute_Value
14	Wapiti	VULNERABILITY_CLASS	Low
14	Wapiti	VULNERABILITY_TYPE	Content Security Policy Configuration
14	Wapiti	INFO	CSP is not set
14	Wapiti	HTTP_REQUEST	GE1
14	Wapiti	SEVERITY_LEVEL	1
14	Wapiti	WSTG	['WSTG-CONF-12', 'OSHP-Content-Security-Policy']
15	Wapiti	VULNERABILITY_CLASS	Low
15	Wapiti	VULNERABILITY_TYPE	Content Security Policy Configuration
15	Wapiti	INFO	CSP is not set
15	Wapiti	HTTP_REQUEST	GET
15	Wapiti	SEVERITY_LEVEL	1
15	Wapiti	WSTG	['WSTG-CONF-12', 'OSHP-Content-Security-Policy']
16	Wapiti	VULNERABILITY_CLASS	Low

146

Figura 25 – Tabela da informação detalhada das vulnerabilidades

5.9.3. Cypress

Uma vez que o Cypress é uma ferramenta distinta das restantes e de forma a ser consistente com a estrutura da base de dados, a sua informação encontra-se numa secção própria do *dashboard*. Esta secção é composta por 2 gráficos circulares (Figura 26) que exibem a informação relativa às *features* e cenários que passaram e falharam nos testes do Cypress, em número absoluto e em percentagem.

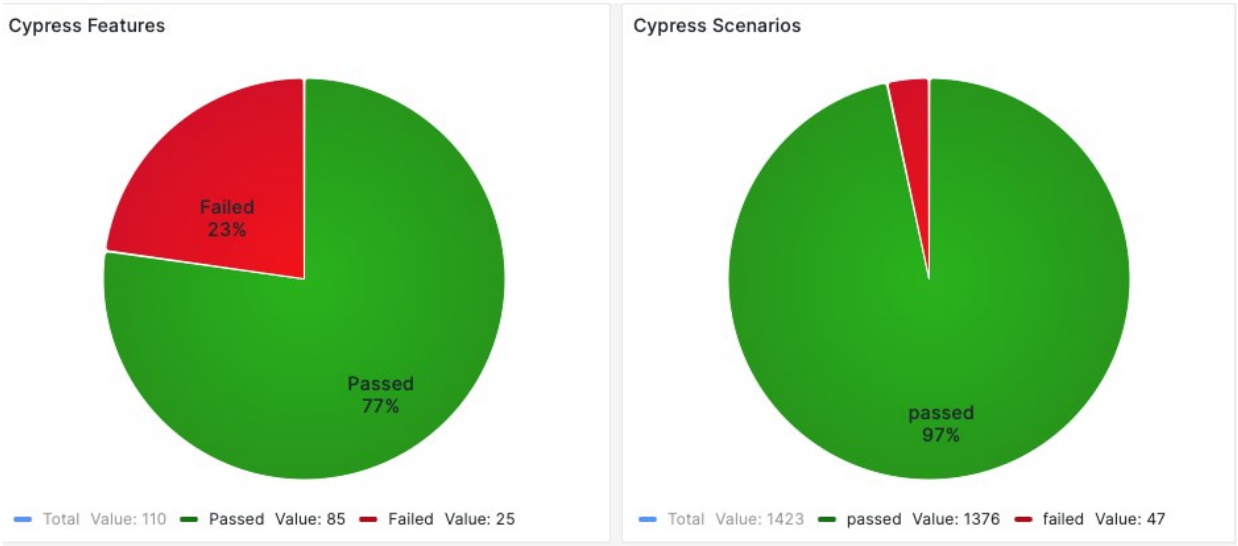


Figura 26 – Gráficos circulares do número e percentagem de *features* e cenários, respetivamente, que passaram e falharam nos testes automáticos

A Figura 27 apresenta o painel seguinte constituído por um gráfico de linhas com a informação do número de *features* da aplicação que foram testadas pelo Cypress ao longo do tempo.

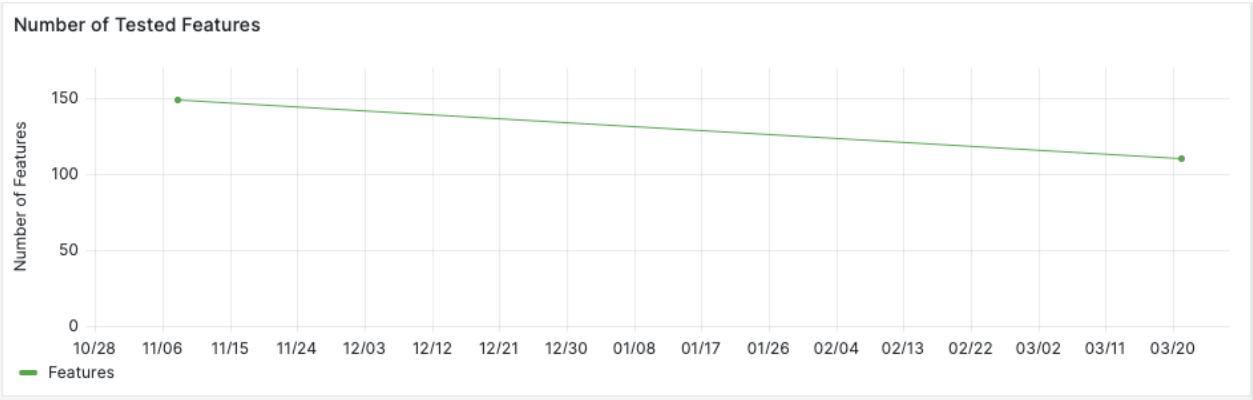


Figura 27 – Gráfico de linhas do número de *features* testadas pelo Cypress ao longo do tempo

O último painel desta secção é composto por uma tabela com o nome da *feature* testada e a duração total dos testes efetuados nessa *feature* (Figura 28). Associado à duração dos testes encontra-se uma barra de medição, que serve como indicador visual do tempo dos testes, i.e. quanto maior, a barra, maior a duração dos testes da *feature* testada. Esta barra possui, ainda, um padrão gradual de cores, com tons de verde para *features* com durações

mais curtas e tons de vermelho ligados a durações de teste maiores. No final da tabela encontra-se a duração total dos testes para todas as *features*.

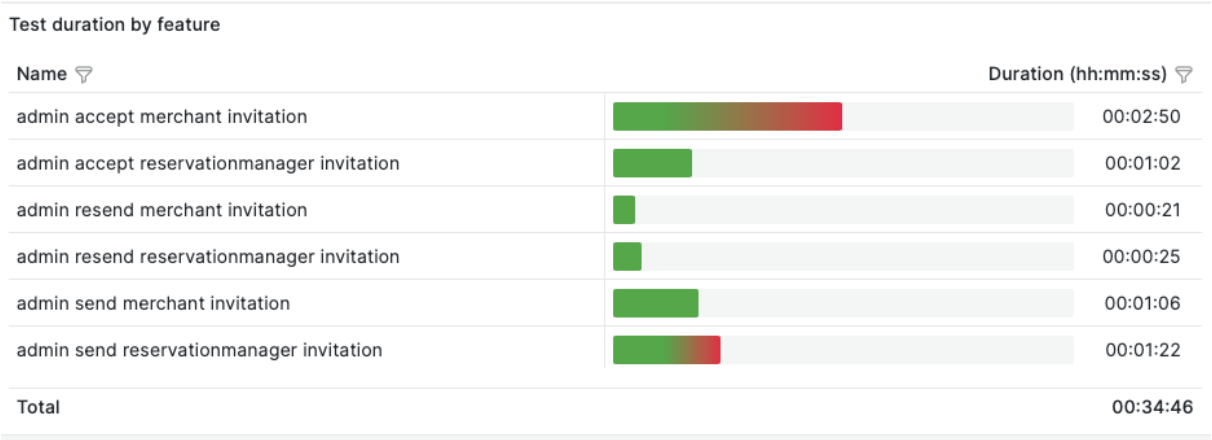


Figura 28 – Tabela do nome e da duração total dos testes das *features* da aplicação

Nesta secção, para além dos painéis, foram ainda adicionados dois *links* (Figura 29) que redirecionam o utilizador para os documentos html originais do Cypress, um do *report* geral dos resultados dos testes e outro com a informação mais detalhada dos testes em cada *step*, permitindo o acesso à informação que não se encontra exibida no *dashboard*. Estas ligações assumem a função de redirecionar o utilizador para uma vista mais detalhada, complementando, dessa forma, a informação visualmente disponível no *dashboard*, tornando a solução mais simples, mais fácil de utilizar e menos confuso para o utilizador, ao mesmo tempo que mantém o seu poder informativo.

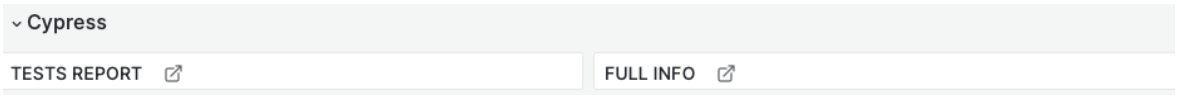


Figura 29 – Hiperligações para os documentos originais gerados pelo Cypress

5.9.4. Filtros

Foram criadas 3 variáveis dinâmicas, utilizando as ferramentas (*scanners*) de segurança, as classes de severidade das vulnerabilidades e a data do *report* de segurança. Estas variáveis são aplicadas aos vários painéis do *dashboard*.

Os primeiros 2 filtros mencionados têm como particularidade a seleção múltipla, i.e., permitem a seleção de mais do que 1 ferramenta ou classe de severidade em simultâneo. A data serve como filtro temporal e possui intervalos de tempo já pré-definidos pelo Grafana, no entanto, é possível inserir intervalos específicos para filtrar a informação.

Outra propriedade importante destes filtros é a possibilidade de poderem ser utilizados em simultâneo, permitindo analisar dos dados de forma mais flexível, especifica e aprofundada. O filtro do *scanner* e da classe de severidade encontram-se ilustrado na Figura 30, enquanto o filtro temporal encontra-se representado na Figura 31.

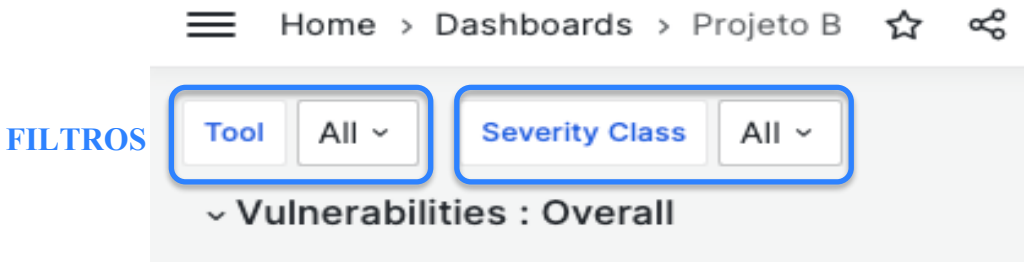


Figura 30 – Filtro de scanners (*Tools)* e de classes de severidade (*Severity Class*) do dashboard

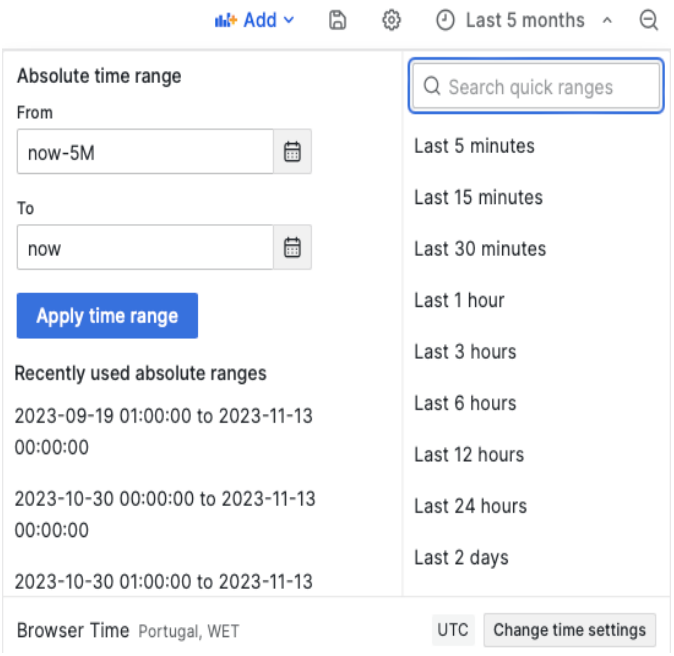


Figura 31 – Filtro temporal do *dashboard*

5.10. Validação da solução

Após o desenvolvimento da solução, foi necessário proceder à sua validação. Esta secção detalha o método e o procedimento empregues neste propósito.

5.10.1. Método de validação

A solução foi validada com recurso a um inquérito de *System Usability Scale* (SUS) [96]. Este questionário permite ter uma visão global da usabilidade de um sistema [97]. Embora os resultados obtidos pelo questionário sejam subjetivos, estudos prévios provaram a sua confiabilidade [98, 99]. Tornou-se um dos métodos mais populares, sendo utilizado em mais de 40% dos estudos de avaliação de sistemas industriais [97, 100]. O questionário é composto por 10 itens simples dispostos numa escala de *Likert*, alternando entre questões de tom positivo (itens ímpares) e questões de tom negativo (itens pares) [100].

De modo a obter as contribuições individuais de cada item para a pontuação, as respostas do questionário são ajustadas através de uma fórmula: para questões de tom positivo a pontuação é calculada subtraindo 1 ao valor da resposta e multiplicando esse valor por 2.5 (por exemplo uma questão positiva com resposta 4 a pontuação será de (4-1)*2.5= 7.5 pontos). Para itens de tom negativo a pontuação é calculada subtraindo o valor da resposta a 5 e multiplicando esse valor por 2.5 (utilizando o exemplo anterior (5-4)*2.5=2.5 pontos) [100]. A pontuação global do sistema é obtida pela soma de todas as pontuações individuais, obtendo, desta forma, um valor numa escala que varia entre 0 e 100, com valores superiores a representar níveis elevados de usabilidade.

5.10.2. Metodologia

A solução foi implementada na infraestrutura da empresa e disponibilizada aos colaboradores dos departamentos de segurança e de QA, que representam os perfis alvo de utilizador, de forma a realizarem uma avaliação. Aos colaboradores foi dada total liberdade para testar todas as funcionalidades, observar e analisar toda a informação disponibilizada pelas visualizações incluídas no *dashboard*, sem qualquer intervenção ou presença do autor deste trabalho, de forma a não influenciar o avaliador.

Após a fase de experimentação e testes, foi fornecido a estes colaboradores um documento composto por uma secção com o questionário SUS e uma secção com três questões de resposta aberta: "funcionalidades que considerou úteis", "limitações" e "funcionalidades inexistentes que considera ser importantes de serem implementadas". Esta segunda secção tinha como intuito obter um feedback mais detalhado e específico dos avaliadores. Um exemplar do documento encontra-se no Anexo G. No total, nove colaboradores preencheram o inquérito, quatro pertencentes ao departamento de segurança e cinco engenheiros de QA.

5.10.3. Resultados

A Tabela 2 apresenta os resultados obtidos no questionário SUS, incluindo a pontuação global e de as pontuações ajustadas de cada item por avaliador, bem como as respetivas pontuações médias (AVG). Os colaboradores foram codificados de acordo com a sua função, em que AS representa os colaboradores de segurança e QA os colaboradores do departamento de QA

Tabela 2 – Pontuações obtidas no questionário SUS

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	TOTAL
AS1	2,5	10	7,5	10	5	5	7,5	2,5	2,5	10	62,5
AS2	7,5	10	0	10	0	10	7,5	10	7,5	10	72,5
AS3	7,5	10	10	10	10	10	10	10	7,5	10	95
AS4	5	10	7,5	10	7,5	10	10	10	5	10	85
QA1	10	5	7,5	10	7,5	7,5	7,5	7,5	7,5	10	80
QA2	5	10	10	10	10	10	7,5	10	7,5	10	90
QA3	5	2,5	7,5	10	7,5	7,5	5	10	5	10	70
QA4	7,5	10	10	10	7,5	10	10	10	7,5	7,5	90

QA5	7,5	10	7,5	10	7,5	7,5	7,5	10	7,5	10	85
AVG	6,4	8,6	7,5	10	6,9	8,6	8,1	8,9	6,4	9,7	81,1

A análise das pontuações médias de cada item revela que todos os colaboradores atribuíram a pontuação máxima ao quarto item (Q4), indicando que não seria necessário apoio técnico para utilizar a solução. A elevada pontuação média dos itens 2 e 8 e 10 sugere que o sistema tem a complexidade adequada ao seu propósito e as suas funcionalidades são facilmente utilizadas pelos utilizadores, embora na perspetiva de dois engenheiros de QA a solução apresente alguma complexidade desnecessária.

O item em que os colaboradores avaliam diretamente a facilidade de uso (Q3) apresentou uma pontuação média inferior aos itens mencionados anteriormente. Tal deve-se ao facto de um dos colaboradores ter atribuído 0 pontos nesta questão, divergindo significativamente das pontuações atribuídas pelos demais colaboradores, podendo, por isso, ser considerado um *outlier*. Os colaboradores consideraram ainda que o sistema não apresenta demasiada inconsistência (Q6) e que a maioria dos utilizadores aprenderia a utilizar o sistema rapidamente.

Do lado menos positivo destacam-se os itens 1 e 9, que obtiveram em média uma pontuação mais baixa. Desta forma, o interesse demonstrado pelos colaboradores em utilizar a solução de forma frequente foi relativamente baixo, bem como a confiança sentida na utilização do sistema. Apesar de apresentar uma pontuação média menor que a maioria dos restantes, sete dos nove colaboradores consideraram que as funcionalidades do sistema estavam maioritariamente bem integradas, pelo que o seu valor médio é influenciado pela pontuação atribuída especialmente pelo AS2, que atribuiu 0 pontos neste item (Q5).

Considerando a média da pontuação global (81,1) a solução apresenta, de forma geral e de acordo com a perspetiva dos avaliadores, um bom nível de usabilidade, não obstante ainda ser um protótipo. Trabalhos anteriores desenvolveram diversas normas para interpretar da melhor forma a pontuação global obtida. Algumas dessas normas incluem o uso de escalas adjetivais e absolutas [98], outras são o resultado de transformações logarítmicas, de forma a normalizar os resultados, e associar as classificações atribuídas aos percentis da distribuição de pontuações, criando uma escala de classificação curva [101].

Utilizando as normas mencionadas e a média da pontuação global, a solução obtém a seguinte classificação:

- Escala adjetival: *Good*
- Escala absoluta: B
- Escala curva: A

As classificações atribuídas por estas escalas à média da pontuação global obtida permitem validar o bom nível de usabilidade da solução.

Na segunda secção do questionário, na questão de "funcionalidades úteis", os colaboradores consideraram os gráficos existentes e a forma de apresentação da informação na secção do Cypress, em especial o gráfico de linhas com o número de *features* testadas ao longo do tempo e a tabela com a duração dos testes por *feature*, uma vez que estes dados não vêm explicitamente apresentados no relatório de testes automáticos que é gerado, complementando a informação existente.

Outras funcionalidades consideradas úteis foram a apresentação do número total de vulnerabilidades detetadas, a opção de filtrar as vulnerabilidades por ferramenta, o gráfico de barras que separa as vulnerabilidades por classe de severidade e o gráfico de linhas com a evolução do número de vulnerabilidades ao longo do tempo.

Como limitações os colaboradores mencionaram a secção de detalhes das vulnerabilidades, especificamente a forma como a informação apresentada se encontra estruturada, tornando-a difícil de ler e analisar os dados presentes.

Um dos colaboradores considerou como limitação o facto do *dashboard* não ter uma secção que apresente "100 % dos dados existentes nos relatórios do Cypress", enquanto que outro colaborador considerou como limitação, a complexidade da secção do Cypress, mencionando que existem relatórios mais básicos "que são mais úteis, devido à sua simplicidade". Foi ainda mencionado que na tabela de duração dos testes por *feature*, que a opção de filtrar os dados por valores não estava a funcionar adequadamente.

Como "funcionalidades inexistentes que considera ser importantes de serem implementadas" foi referida a necessidade de existir um gráfico que mostrasse a evolução do número de *features* e *scenarios* que passam e que falham ao longo do tempo e presença

de uma coluna adicional na tabela de duração dos testes por *feature*, com a informação se a *feature* passou ou falhou, uma vez que isto permitiria associar a passagem ou falha de uma *feature* nos testes à sua duração.

Adicionalmente, considerou-se altamente benéfica a existência de uma funcionalidade que permita gerar um relatório com toda a informação apresentada, para entregar aos clientes da empresa. Um dos colaboradores sugeriu a possibilidade da presença de um *popup* nos gráficos circulares do Cypress, em que através de um clique numa das fatias do gráfico, apresentasse uma lista ou tabela de todas *features* ou *scenarios* daquele estado (por exemplo, caso se clique na fatia dos testes que falharam no gráfico circular das *features*, era apresentada uma lista com todas as *features* que falharam).

6. Análise da solução desenvolvida

Neste capítulo é realizada uma análise crítica aprofundada de todas as decisões, procedimentos e abordagens adotadas ao longo de todo o desenvolvimento, bem como da solução final desenvolvida. Serão ainda abordados os principais desafios e constrangimentos encontrados durante a realização das várias tarefas. É efetuada uma análise comparativa entre a solução desenvolvida e as soluções identificadas na secção 3.2.

6.1. Análise global da arquitetura

A arquitetura projetada cumpre os requisitos de funcionalidade necessários na medida em que permite processar adequadamente os dados provenientes dos *scanners* de segurança, transformando-os em *insights* relevantes que servirão de base para a tomada de decisão futura no que concerne à segurança das aplicações.

Os pontos fortes desta arquitetura residem na sua flexibilidade e escalabilidade, permitindo adicionar facilmente novos atributos e incorporar novos *scanners* de segurança como fontes de dados. O grau de complexidade da solução não é excessivamente alto, facilitando a sua implementação e manutenção. O nível elevado de automatização garante um desempenho elevado, reduzindo bastante o erro humano, a um tempo de execução mais rápido em comparação com uma arquitetura mais dependente de processos manuais.

Não obstante, a arquitetura apresenta ainda algumas limitações no processo inicial de extração e carregamento dos dados na base de dados. Estas limitações são exploradas nas secções seguintes onde são analisados os vários processos que compõem a solução desenvolvida.

6.1.1. Extração de dados

O principal desafio nesta etapa foi desenvolver uma função de extração que tivesse em consideração o formato dos relatórios, a estrutura e tipo de dados. O *script* de extração de dados atual tem uma função adicional de filtro de dados inicial, uma vez que apenas extrai os dados considerados relevantes para o *dashboard*. Um dos problemas que persistem

prende-se não com o processo de extração em si, mas com o facto de os *outputs* dos *scanners*, que servem de *input* para o *script* terem de ser descarregados e guardados manualmente.

6.1.2. Transformação dos dados

O *script* de transformação constitui uma etapa essencial no processo ETL, reestruturando os dados de forma a coincidir com a estrutura da base de dados relacional, onde todos os valores dos vários são agrupados num único campo (*Atribute_Value*). É também nesta etapa que é adicionada a coluna da data que constituirá a dimensão temporal, estritamente necessária para alcançar o objetivo da solução. A primeira versão deste *script* tinha a limitação de ser necessário alterar manualmente a data todas as vezes em que o *script* é executado, contundo, a versão atual já considera, de forma automática, a data de execução do *script*.

6.1.3. Modelo e Base de dados

Como mencionado na secção 5.7, a maior desafio foi a construção de um modelo de dados que fosse suficientemente flexível e escalável para permitir a incorporação dos vários tipos de dados extraídos, bem como adicionar futuramente, nova informação e novas ferramentas de segurança à base de dados. Este objetivo é conseguido através do agrupamento de todos os dados extraídos num único campo (*Atribute_Value*). As tabelas permitem catalogar a informação extraída *Project*, *Atribute*, *Tool* e *Vulnerability* e permitem catalogar a origem dos dados extraídos através do uso de chaves estrangeiras.

6.1.4. Importação dos dados

Inicialmente foram criadas *tasks* no DBeaver que permitem a importação semiautomática dos dados. Com o escalar da solução para todos os projetos da empresa, a importação dos dados através das *tasks* torna-se moroso e por isso inviável. O *script* de transformação de dados sofreu alterações que vão de encontro à importação dos dados de forma automática, no futuro.

6.1.5. *Dashboard* de segurança de aplicações

A solução desenvolvida cumpre os requisitos definidos inicialmente. O *dashboard* permite uma visão holística, integrada e detalhada do estado de segurança das aplicações, alinhado com os objetivos e necessidades da empresa. A sua estruturação em três diferentes secções, facilita o acesso por parte dos utilizadores, à informação que necessitam para a sua função, minimizando o ruído e evitando uma sobrecarga de informação. Em cada secção, os painéis estão organizados de forma lógica e clara e apresentando a informação de forma sucinta e encadeada, facilitando a análise e compreensão dos dados disponibilizados.

Os resultados obtidos dos inquéritos SUS e o feedback proporcionado pelos colaboradores vão de encontro a estas afirmações. A secção relativa ao Cypress foi particularmente destacada pelos colaboradores de forma positiva, onde consideraram como funcionalidades úteis os gráficos existentes e a forma de como a informação é apresentada nessa secção. Os inquéritos também permitiram concluir que na perspetiva dos utilizadores, a solução não é demasiado complexa, não sendo necessário apoio técnico, nem muito tempo de aprendizagem, para os vários tipos de utilizadores usarem o sistema corretamente.

A secção de detalhes sob demanda permite o acesso rápido a informação mais detalhada das vulnerabilidades identificadas sempre que necessário. Esta estrutura de organização da informação garante um elevado nível de granularidade dos dados. Esta secção apresenta, porém, uma grande limitação na forma em que a informação detalhada das vulnerabilidades é apresentada ao utilizador.

A incorporação da dimensão temporal na análise das vulnerabilidades viabiliza a avaliação e monitorização de forma contínua do estado global das aplicações ao nível da segurança. Por sua vez, a monitorização constante em conjunto com a análise mais profunda contribui para a tomada de decisões mais informadas e consequentemente à adoção de procedimentos e respostas de segurança mais eficientes.

A existência de um filtro temporal é também de especial utilidade, permitindo identificar tendências, comparar o estado da aplicação em diferentes períodos temporais ou na análise de todo o histórico de vulnerabilidades. Na secção de *overview* das vulnerabilidades, o gráfico de linhas que incorpora esta dimensão foi mencionado como funcionalidade útil, validando, deste modo, a utilidade desta dimensão na monitorização do estado de segurança das aplicações.

Os filtros dinâmicos, mencionados pelos colaboradores com funcionalidade útil, facilitam a análise e interpretação dos dados ao possibilitar a exploração dos dados de forma interativa, aumentando o desempenho e utilidade do *dashboard*. Apesar de tudo, existem limitações na interatividade da solução, associados ao programa de visualização de dados utilizado (Grafana), que não permite atualmente a opção de implementar *cross-filtering* ou da criação de *pop-ups*, que permitiriam um nível superior de interatividade entre as visualizações e o utilizador.

6.2. Análise comparativa

A Tabela 3 apresenta um resumo comparativo das ferramentas mencionadas na secção 3.2**,** identificadas pelas suas respetivas referências bibliográficas e da solução desenvolvida, destacando as funcionalidades chave de cada uma, bem como as principais semelhanças e diferenças entre elas.

Com exceção da última, as funcionalidades presentes em cada ferramenta estão assinaladas com um "X" enquanto a ausência da funcionalidade se encontra representada por um "–". Importa ainda mencionar que não foi possível determinar a presença ou ausência de certas funcionalidades para algumas das ferramentas. Estes casos encontram-se indicados na tabela pela conotação "N/D" (Não Determinado). A última funcionalidade é relativa à utilização de métricas por parte da ferramenta. Após ponderação, concluiu-se que a melhor forma de comparar as ferramentas relativamente a métricas não eram a sua presença ou ausência, mas sim a complexidade das métricas exibidas pelas várias visualizações.

Tabela 3 – Tabela comparativa das soluções existentes e o protótipo desenvolvido

Funcionalidade/ Ferramenta	[64]	[65]	[66]	[67]	[7]	**Solução desenvolvida**
Filtros	X	X	X	X	X	X
Detalhes sob demanda	X	X	X	X	X	X

Cross-filtering	–	N/D	X	N/D	N/D	–
Múltiplos filtros em simultâneo	–	–	X	–	X	X
Associação de cores à representação dos dados	–	X[1]	X	–	X	X
Visualização do número geral de vulnerabilidades	–	X	–	–	N/D	X
Vista de comparação de relatórios	–	X[1]	–	–	X	–
Comparação de dados	–	X	X	X	X	X
Zooming	X	–	X	X	X	X
Incorporação de dados temporais	–	–	X	X	X[2]	X
Compatibilidade com diferentes scanners	X	X	X[3]	X	X	X

[1] Apesar de estar presente, foram identificados problemas na funcionalidade

[2] Não utiliza dados temporais diretamente

[3] Embora seja compatível com vários scanners, em alguns casos, a informação das vulnerabilidades encontradas não está disponível e como tal não é exibida

Compatibilidade com diferentes tipos de scanners	–	–	–	–	–	X
Função de pesquisa	–	X	–	N/D	N/D	X
Presença de métricas	–	Simples	Simples	Simples	Avançadas	Simples

Todas as soluções existentes incorporam funcionalidades consideradas fundamentais numa ferramenta de visualização de dados, como filtragem e uma secção dedicada à apresentação de detalhes sob demanda. Não obstante, foram identificados problemas e /ou limitações nas ferramentas existentes que em comparação, a solução desenvolvida neste trabalho não apresenta.

Três das soluções [64, 65, 67] não apresentam a funcionalidade que permite ao utilizador filtrar a informação através de múltiplos filtros. A ausência desta funcionalidade limita a capacidade exploratória proporcionada pela ferramenta e a sua interatividade. Para além disso, estas ferramentas falham na utilização de cores para representar os dados [64, 67] ou utilizam cores inadequadamente [65].

Com base na informação apresentada na Tabela 3, a ferramenta Cesar [64] pode ser considerada mais básica e limitada em termos de visualizações e funcionalidades oferecidas ao utilizador. A apresentação da informação restringe-se a um único *treemap*, comprometendo a sua adequação para todos os tipos de análise de dados e limitando, dessa forma, a sua eficácia em representar os diferentes aspetos relacionados com a segurança de aplicações. Para além disso, de todas as ferramentas mencionadas, é a única que não oferece ao utilizador nenhum tipo de visualização com capacidade comparativa de dados.

A ferramenta concebida em [65] proporciona uma visualização detalhada de um único relatório de vulnerabilidades. É a única, para além da solução criada neste trabalho, que exibe o número absoluto de vulnerabilidades identificadas no relatório e que menciona ter a funcionalidade de pesquisa de vulnerabilidades. Uma das funcionalidades mais distintas

desta ferramenta, apenas presente numa das restantes, consiste na possibilidade dos utilizadores compararem diferentes versões da aplicação em termos de vulnerabilidades, através de uma vista comparativa de dois relatórios diferentes. Esta opção permite uma análise evolutiva da segurança da aplicação, no entanto, durante a fase de avaliação da ferramenta, esta funcionalidade apresentou problemas em fazer o *match* das vulnerabilidades presentes nos dois relatórios.

A solução desenvolvida em [66] destaca-se das anteriores pela sua interatividade, com três níveis de granularidade dos dados, proporcionando assim uma análise interativa e detalhada e sendo a única que possui a funcionalidade de *cross-filtering*. Esta ferramenta combina a informação de *scanners* SAST com informação de proveniência do código fonte e que através de *zooming*, possibilita aos utilizadores analisar eventos de desenvolvimento de *software* com interesse de forma isolada. A principal limitação da ferramenta reside no fato de que a informação é agrupada por categoria CWE e esse atributo não está acessível para todas as possíveis vulnerabilidades, resultando numa visualização incompleta dos dados.

A primeira solução com foco específico nas vulnerabilidades identificadas pelos *scanners* DAST [67], utiliza a estrutura hierárquica do website em conjunto com os resultados dos scans de vulnerabilidades, para fornecer e exibir graficamente estatísticas da informação obtida. Os resultados de vários *scanners* DAST sofrem um processo de normalização. A informação encontra-se disposta num formato de *rooted-tree* de forma a manter a estrutura da página *web* analisada. Esta técnica de visualização apresenta uma grande limitação em termos de desempenho: em aplicações de grande dimensão com mais de 70 nódulos, torna-se árduo para o utilizador analisar e interpretar a informação apresentada.

De todas as soluções desenvolvidas em trabalhos prévios, a solução concebida em [7] aparenta ter maior robustez, com um grande número de funcionalidades. Uma característica diferenciadora desta ferramenta é a utilização de uma grande quantidade de métricas, mais avançadas nas suas visualizações do *dashboard* para abordar o estado de segurança das aplicações, superando as restantes soluções, incluído a desenvolvida neste trabalho, que apresentam um conjunto limitado de métricas mais simples, maioritariamente baseadas em estatísticas.

Embora ainda se trate da versão inicial de um protótipo, a análise comparativa realizada demonstra que a solução desenvolvida neste trabalho já abrange a maioria das funcionalidades presentes em ferramentas semelhantes, chegando mesmo a apresentar capacidades adicionais, comparativamente com alguns casos.

O fator diferenciador desta solução é a sua capacidade de integração e utilização de dados de múltiplos tipos de ferramentas de segurança e apresentá-los num único *dashboard*, obtendo uma visão holística. É esta capacidade que aliada à incorporação de dados temporais, permite uma monitorização e análise adequada da evolução do estado de segurança das várias aplicações durante todo o seu período de desenvolvimento. Este aspeto inovador caracteriza a unicidade da solução desenvolvida, destacando-a das soluções identificadas na secção 3.2.

7. Conclusão e Trabalho Futuro

O número de atividades de cibercrime que exploram as vulnerabilidades existentes em aplicações *Web* tem sofrido um crescimento notável nos últimos anos e segundo as últimas previsões, essa tendência deverá continuar. Este fenómeno intensificou a necessidade das várias organizações fortalecerem os seus protocolos e políticas de segurança aplicadas durante o SDLC. No entanto, no âmbito da segurança de aplicações, a visualização destes dados permanece uma área bastante inexplorada, onde o estado atual da arte se foca na apresentação de dados de um único tipo de ferramenta de segurança.

O presente trabalho teve como foco o desenvolvimento de uma solução que permite a visualização de dados de segurança das aplicações *Web* desenvolvidas pela empresa, utilizando como fonte de dados os relatórios gerados por diferentes tipos ferramentas de segurança e armazenando dados temporais relativos às vulnerabilidades identificadas. Foi desenhada uma arquitetura que permite executar com sucesso todos os processos, desde a obtenção dos dados dos relatórios das ferramentas de segurança até à sua visualização num *dashboard* integrado.

A solução desenvolvida permite alcançar os objetivos estabelecidos inicialmente para este trabalho, na medida em que as funcionalidades apresentadas proporcionam ao utilizador uma visão holística do estado atual de segurança da aplicação e, adicionalmente, incorporam a dimensão temporal para a análise e o monitoramento contínuo desse estado ao longo do tempo. Esta visão integrada e abrangente do estado de segurança das aplicações possibilita a tomada de decisões mais informadas e baseadas em dados, promovendo a implementação de melhores práticas de segurança no SDLC.

Apesar de ainda se encontrar na sua primeira versão, os resultados obtidos nos questionários SUS indicam que a solução já apresenta um bom nível de usabilidade. A análise comparativa demonstra, também, que a solução abrange a maioria, ou até mais, das funcionalidades encontradas em ferramentas semelhantes, destacando-se pela sua capacidade de integração e apresentação de dados de múltiplos tipos de ferramentas de segurança num *dashboard* unificado.

O trabalho futuro consiste em abordar as limitações e pontos de melhoria já identificados, que não foram possíveis de corrigir devido a restrições temporais, bem como automatizar

completamente os vários processos, tais como: o acesso e extração dos dados, sem necessidade de descarregar manualmente os relatórios das ferramentas de segurança, através de *requests* às várias API; a importação dos dados para a MariaDB, utilizando uma biblioteca *sql* no *script* que permite a conexão direta à base de dados; e executar os vários *scripts* automaticamente quando necessário. Tal automatização pode ser alcançada através do agendamento da execução dos *scripts*, através da configuração de um *cron job* [4]. O feedback obtido nos inquéritos SUS será analisado e, caso seja possível, incorporado numa versão futura da solução, promovendo um processo iterativo de melhoria e de feedback contínuo.

Será feito um levantamento de potenciais métricas a incluir no *dashboard*, utilizando como base a lista de métricas utilizadas em [7]. Essas métricas serão analisadas individualmente de modo a perceber a sua utilidade no contexto dos dados disponíveis nos painéis de visualização. Note-se que este é um ponto bastante importante, pois, como referido anteriormente, o número reduzido e grau elevado de simplicidade das métricas utilizadas constitui atualmente uma limitação da solução desenvolvida.

Por fim, seria igualmente importante para o desenvolvimento da solução tirar partido do modelo de dados e adicionar novos tipos de dados relacionados com a segurança das aplicações e/ou informação associada às fases de desenvolvimento, permitindo que as vulnerabilidades sejam associadas a eventos específicos do processo de desenvolvimento da aplicação.

[4] Tarefa que permite agendar e automatizar processos em sistemas operacionais baseados em *Unix*

Bibliografia

[1] "Escola Superior de Tecnologia e Gestão." Accessed: Sep. 15, 2023. [Online]. Available: https://www.ipleiria.pt/estg/

[2] "Politécnico de Leiria." Accessed: Sep. 15, 2023. [Online]. Available: https://www.ipleiria.pt/

[3] "VOID Software :: the software specialists." Accessed: Sep. 15, 2023. [Online]. Available: https://void.pt/

[4] P. Langlois, A. Pinto, D. Hylender, and S. Widup, "DBIR 2023 Data Breach Investigations Report 10K 20K 30K About the cover," Jun. 2023. doi: 10.13140/RG.2.2.32362.70085.

[5] H. Myrbakken and R. Colomo-Palacios, "DevSecOps: A multivocal literature review," in *Software Process Improvement and Capability Determination: 17th International Conference*, Springer International Publishing, Sep. 2017, pp. 17–29. doi: 10.1007/978-3-319-67383-7_2.

[6] Xffccdd, "DevSecOps Definition, Best Practices and Tools," Medium. Accessed: Nov. 30, 2023. [Online]. Available: https://medium.com/@cloud_tips/devsecops-definition-best-practices-and-tools-1789587d165a

[7] F. O. Sonmez and B. G. Kilic, "Holistic Web Application Security Visualization for Multi-Project and Multi-Phase Dynamic Application Security Test Results," *IEEE Access*, vol. 9, 2021, doi: 10.1109/ACCESS.2021.3057044.

[8] *V. Software*, Documento Interno, 2023, *VOID Software*.

[9] "UptoData." Accessed: Dec. 22, 2024. [Online]. Available: https://uptodata.pt/

[10] A. Penela, B. Sequeira, E. Piza, M. B. Piedade, R. Matias, and C. I. Reis, "Security Insights at a Glance: An Integrated Data Approach for Application Security," in *2024 IEEE 18th International Conference on Application of Information and Communication Technologies (AICT)*, 2024, pp. 1–6. doi: 10.1109/AICT61888.2024.10740411.

[11] "AICT2024 | The 18th IEEE International Conference on Application of Information and Communication Technologies." Accessed: Dec. 22, 2024. [Online]. Available: https://www.aict.info/index.php?csc=2024

[12] A. E. Shadare, S. M. Musa, C. Akujuobi, M. N. O. Sadiku, C. M. Akujuobi, and R. G. Perry, "DATA VISUALIZATION," *International Journal of Engineering Research And Advanced Technology*, 2016, [Online]. Available: https://www.researchgate.net/publication/311597028

[13] E. F. Sinar, "Data Visualization," in *Big Data at Work*, Routledge, 2015, pp. 115–157.

[14] M. Engebretsen and H. Kennedy, *Data Visualization in Society*. Amsterdam: Amsterdam University Press, 2020.

[15] M. Aparicio and C. J. Costa, "Data visualization," *Communication Design Quarterly*, vol. 3, no. 1, pp. 7–11, Jan. 2015, doi: 10.1145/2721882.2721883.

[16] S. A. Keller, S. S. Shipp, A. D. Schroeder, and G. Korkmaz, "Doing Data Science: A Framework and Case Study," *Harv Data Sci Rev*, vol. 2, no. 1, p. 2020, Jan. 2020, doi: 10.1162/99608F92.2D83F7F5.

[17] L. Veldkamp, "Valuing Data as an Asset," *Rev Financ*, vol. 27, no. 5, pp. 1545–1562, Sep. 2023, doi: 10.1093/rof/rfac073.

[18] M. Gupta and J. F. George, "Toward the development of a big data analytics capability," *Information and Management*, vol. 53, no. 8, pp. 1049–1064, Dec. 2016, doi: 10.1016/j.im.2016.07.004.

[19] Gartner, "Gartner Survey Reveals That 64 Percent of Organizations Have Invested or Plan to Invest in Big Data in 2013," 2013.

[20] O. S. Technologies, "Data Visualization: Importance, benefits and techniques | OakStreet Technologies | Medium," Medium. Accessed: Dec. 22, 2024. [Online]. Available: https://medium.com/@oakstreetechnologies/data-visualization-importance-benefits-and-techniques-a26b73cfb6f7

[21] Y. Zhu, "Measuring Effective Data Visualization," *Lecture Notes in Computer Science*, vol. 4842 LNCS, no. PART 2, pp. 652–661, 2007, doi: 10.1007/978-3-540-76856-2_64.

[22] Q. Li, *Embodying data: Chinese aesthetics, interactive visualization and gaming technologies*. 2020. doi: 10.1007/978-981-15-5069-0.

[23] Andy. Kirk, *Data visualization : a successful design process* . Birmingham, UK: Packt Pub., 2012.

[24] L. Ticong, "Common Data Visualization Examples: Transform Numbers into Narratives." Accessed: Dec. 22, 2024. [Online]. Available: https://www.datamation.com/big-data/data-visualization-use-cases/

[25] A. Abduldaem and A. Gravell, "Principles for the design and development of dashboard: Literature review," *INTCESS 2019- 6th International Conference on Education and Social Sciences*, no. February, 2019.

[26] A. A. Rahman, Y. B. Adamu, and P. Harun, "Review on dashboard application from managerial perspective," in *International Conference on Research and Innovation in Information Systems, ICRIIS*, 2017. doi: 10.1109/ICRIIS.2017.8002461.

[27] S. Few, "Dashboard Confusion Revisited," *Perceptual Edge*, 2007.

[28] O. M. Yigitbasioglu and O. Velcu, "A review of dashboards in performance management: Implications for design and research," *International Journal of Accounting Information Systems*, vol. 13, no. 1, 2012, doi: 10.1016/j.accinf.2011.08.002.

[29] W. Noonpakdee, T. Khunkornsiri, A. Phothichai, and K. Danaisawat, "A framework for analyzing and developing dashboard templates for small and medium enterprises," in *2018 5th International Conference on Industrial Engineering and Applications, ICIEA 2018*, 2018. doi: 10.1109/IEA.2018.8387148.

[30] S. H. Dolatabadi, E. Gatial, I. Budinska, and Z. Balogh, "Integrating Human-Computer Interaction Principles in User-Centered Dashboard Design: Insights from Maintenance Management," in *INES 2024 - 28th IEEE International Conference on Intelligent Engineering Systems 2024, Proceedings*, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 219–224. doi: 10.1109/INES63318.2024.10629098.

[31] A. P. Chokki, A. Simonofski, B. Frénay, and B. Vanderose, "Engaging Citizens with Open Government Data: The Value of Dashboards Compared to Individual Visualizations," *Digital Government: Research and Practice*, vol. 3, no. 3, 2022, doi: 10.1145/3558099.

[32] A. Andy, A. John, and K. Chris, “From Data to Insight: A Framework for Real-Time Cybersecurity Analytics and Visualization,” Jan. 2025. [Online]. Available: https://www.researchgate.net/publication/388195117

[33] “Dashboard Development: Case-based Guide.” Accessed: Mar. 12, 2025. [Online]. Available: https://maybe.works/blogs/dashboard-development

[34] K. Pillai, “7-Step Process to Build a Dashboard .” Accessed: Mar. 13, 2025. [Online]. Available: https://www.linkedin.com/pulse/7-step-process-build-dashboard-krishnadev-pillai/

[35] “Learn 25 Dashboard Design Principles & BI Best Practices,” RIB Software. Accessed: Mar. 12, 2025. [Online]. Available: https://www.rib-software.com/en/blogs/bi-dashboard-design-principles-best-practices

[36] V. M., “A project framework for effective dashboards ,” Medium. Accessed: Mar. 12, 2025. [Online]. Available: https://medium.com/p-metrics/a-dashboarding-framework-for-actionable-insights-58d819ceccc0

[37] A. C. Bock and U. Frank, “Low-Code Platform,” *Business and Information Systems Engineering*, vol. 63, no. 6, 2021, doi: 10.1007/s12599-021-00726-8.

[38] “What Is Low-Code? ,” IBM. Accessed: Mar. 13, 2025. [Online]. Available: https://www.ibm.com/think/topics/low-code

[39] V. K. Srirangam and R. K. Kommidi, “DEMOCRATIZING DATA INSIGHTS: THE IMPACT OF NO-CODE/LOW-CODE PLATFORMS ON BUSINESS INTELLIGENCE VISUALIZATION,” *International Journal of Research In Computer Applications and Information Technology (IJRCAIT)*, vol. 7, no. 2, pp. 647–661, doi: 10.5281/zenodo.14025139.

[40] D. Pinho, A. Aguiar, and V. Amaral, “What about the usability in low-code platforms? A systematic literature review,” *J Comput Lang*, vol. 74, 2023, doi: 10.1016/j.cola.2022.101185.

[41] A. Hoffman, *Web Application Security: Exploitation and Countermeasures for Modern Web Applications*. O’Reilly Media, Inc., 2020.

[42] M. Soegaard, “Native, web or hybrid app: Which one is better?” Accessed: Dec. 12, 2024. [Online]. Available: https://www.interaction-design.org/literature/article/native-vs-hybrid-vs-responsive-what-app-flavour-is-best-for-you

[43] I. S. Education, “O que é uma web App, quem a desenvolve e como ?” Accessed: Nov. 27, 2023. [Online]. Available: https://www.linkedin.com/pulse/o-que-é-uma-web-app-quem-desenvolve-e-como-imf-portugal/?originalSubdomain=pt

[44] William, “Web Application Architecture: The Latest Guide for 2024,” ClickIT. [Online]. Available: https://www.clickittech.com/devops/web-application-architecture/

[45] F. Ramos *et al.*, “Vulnerabilidades em aplicações web,” *RE3C-Revista Eletrônica Científica de Ciência da Computação*, vol. 8, no. 1, 2013.

[46] “3 Tier Architecture.” Accessed: Nov. 27, 2023. [Online]. Available: https://www.sketchbubble.com/en/presentation-3-tier-architecture.html

[47] M. Alenezi and Y. Javed, “Open source web application security: A static analysis approach,” *2016 International Conference on Engineering and MIS, ICEMIS 2016*, Nov. 2016, doi: 10.1109/ICEMIS.2016.7745369.

[48] “OWASP Top 10,” OWASP. Accessed: Apr. 04, 2024. [Online]. Available: https://owasp.org/Top10/

[49] S. Chacon and B. Straub, “Git on the Server – Setting Up the Server,” in *Pro Git*, 2nd ed., Apress, 2014, ch. Git on the Server.

[50] B. Jammeh, “DevSecOps: Security Expertise a Key to Automated Testing in CI/CD Pipeline.”

[51] T. Rangnau, R. V. Buijtenen, F. Fransen, and F. Turkmen, “Continuous Security Testing: A Case Study on Integrating Dynamic Security Testing Tools in CI/CD Pipelines,” in *Proceedings - 2020 IEEE 24th International Enterprise Distributed Object Computing Conference, EDOC 2020*, Institute of Electrical and Electronics Engineers Inc., Oct. 2020, pp. 145–154. doi: 10.1109/EDOC49727.2020.00026.

[52] I. Sacolick, “What is CI/CD? Continuous integration and continuous delivery explained,” InfoWorld. Accessed: Apr. 04, 2024. [Online]. Available:

https://www.infoworld.com/article/2269266/what-is-cicd-continuous-integration-and-continuous-delivery-explained.html

[53] "What Is CI/CD? Definition, Process, Benefits, and Best Practices," Spiceworks Inc. Accessed: Apr. 04, 2024. [Online]. Available: https://www.spiceworks.com/tech/devops/articles/what-is-ci-cd/

[54] M. Wilczek, "Average cost of a data breach in 2020: $3.86M." Accessed: Nov. 30, 2023. [Online]. Available: https://www.darkreading.com/vulnerabilities-threats/average-cost-of-a-data-breach-in-2020-3-86m

[55] M. J. Page *et al.*, "The PRISMA 2020 statement: An updated guideline for reporting systematic reviews," 2021. doi: 10.1136/bmj.n71.

[56] "Google Scholar." Accessed: May 13, 2024. [Online]. Available: https://scholar.google.com/

[57] "ACM Digital Library," ACM Digital Library. Accessed: May 14, 2024. [Online]. Available: https://dl.acm.org/

[58] "IEEE Xplore." Accessed: May 14, 2024. [Online]. Available: https://ieeexplore.ieee.org/Xplore/home.jsp

[59] "Springer - International Publisher Science, Technology, Medicine | Springer — International Publisher." Accessed: May 14, 2024. [Online]. Available: https://www.springer.com/gp

[60] B. Yigit Ozkan, S. van Lingen, and M. Spruit, "The Cybersecurity Focus Area Maturity (CYSFAM) Model," *Journal of Cybersecurity and Privacy*, vol. 1, no. 1, 2021, doi: 10.3390/jcp1010007.

[61] K. S. Tiong Tan, A. S. Hoong Lee, and C. T. Min, "Studying The Perception of Using Visualization Dashboard to Measure Cybersecurity Maturity Stage," in *International Conference on Research and Innovation in Information Systems, ICRIIS*, 2021. doi: 10.1109/ICRIIS53035.2021.9617069.

[62] F. Ö. Sönmez and B. Günel, "Security visualization extended review: issues, classifications, validation methods, trends, extensions," in *Security and Privacy Management, Techniques, and Protocols*, 2018. doi: 10.4018/978-1-5225-5583-4.ch006.

[63] Y. S. Wijaya, "Web-Based Dashboard for Monitoring Penetration Testing Activities Based on OWASP Standards," *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika*, vol. 6, no. 1, 2020, doi: 10.26555/jiteki.v16i1.17019.

[64] H. Assal, S. Chiasson, and R. Biddle, "Cesar: Visual representation of source code vulnerabilities," in *2016 IEEE Symposium on Visualization for Cyber Security, VizSec 2016*, 2016. doi: 10.1109/VIZSEC.2016.7739576.

[65] S. L. Reynolds, T. Mertz, S. Arzt, and J. Kohlhammer, "User-Centered Design of Visualizations for Software Vulnerability Reports," in *Proceedings - 2021 IEEE Symposium on Visualization for Cyber Security, VizSec 2021*, 2021. doi: 10.1109/VizSec53666.2021.00013.

[66] A. Schreiber, T. Sonnekalb, and L. Von Kurnatowski, "Towards Visual Analytics Dashboards for Provenance-driven Static Application Security Testing," in *Proceedings - 2021 IEEE Symposium on Visualization for Cyber Security, VizSec 2021*, 2021. doi: 10.1109/VizSec53666.2021.00010.

[67] T. T. Dang and T. K. Dang, "An extensible framework for web application vulnerabilities visualization and analysis," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8860, 2014, doi: 10.1007/978-3-319-12778-1_7.

[68] J. P. B. F. C. Marques, "Web Security Application Project," Msc Thesis, Instituto Politécnico de Leiria, Leiria, 2021.

[69] Insidersec, "Insider," GitHub. Accessed: Sep. 30, 2023. [Online]. Available: https://github.com/insidersec/insider

[70] K. Kuszczyński and M. Walkowski, "Comparative Analysis of Open-Source Tools for Conducting Static Code Analysis," *Sensors*, vol. 23, no. 18, 2023, doi: 10.3390/s23187978.

[71] N. Surribas, "Wapiti : a Free and Open-Source web-application vulnerability scanner in Python." Accessed: Sep. 30, 2023. [Online]. Available: https://wapiti-scanner.github.io/

[72] A. Al Anhar and Y. Suryanto, "Evaluation of Web Application Vulnerability Scanner for Modern Web Application," in *ICAICST 2021 - 2021 International Conference on Artificial*

Intelligence and Computer Science Technology, 2021. doi: 10.1109/ICAICST53116.2021.9497831.

[73] "OWASP Zed Attack Proxy Project," OWASP. Accessed: Oct. 01, 2023. [Online]. Available: https://www.zaproxy.org/

[74] "Testing Frameworks for Javascript | Write, Run, Debug | Cypress," Cypress. Accessed: Oct. 03, 2023. [Online]. Available: https://www.cypress.io/

[75] C. Solís and X. Wang, "A study of the characteristics of behaviour driven development," in *Proceedings - 37th EUROMICRO Conference on Software Engineering and Advanced Applications, SEAA 2011*, 2011. doi: 10.1109/SEAA.2011.76.

[76] "OWASP Dependency-Track," OWASP Foundation. Accessed: Nov. 07, 2023. [Online]. Available: https://owasp.org/www-project-dependency-track/

[77] J. L. Sánchez, "Dependency Track: Analyze your vulnerabilities from the use of third-party components," Geko Cloud. Accessed: Nov. 08, 2023. [Online]. Available: https://geko.cloud/en/dependency-track-analyze-your-vulnerabilities-from-the-use-of-third-party-components/

[78] "Jenkins." Accessed: Oct. 25, 2023. [Online]. Available: https://www.jenkins.io/

[79] M. Heller, "What is Jenkins? The CI server explained," InfoWorld. Accessed: Oct. 25, 2023. [Online]. Available: https://www.infoworld.com/article/2260091/what-is-jenkins-the-ci-server-explained.html

[80] Saurabh, "What is Jenkins: Jenkins For Continuous Integration," Edureka. Accessed: Oct. 25, 2023. [Online]. Available: https://www.edureka.co/blog/what-is-jenkins/

[81] "DBeaver Community | Free Universal Database Tool." Accessed: Dec. 18, 2023. [Online]. Available: https://dbeaver.io/

[82] Dbeaver, "GitHub - dbeaver/dbeaver: Free universal database tool and SQL client," GitHub. Accessed: Dec. 19, 2023. [Online]. Available: https://github.com/dbeaver/dbeaver

[83] "Top 5 database GUIs for SQL databases 2024," DronaHQ. Accessed: Apr. 14, 2024. [Online]. Available: https://www.dronahq.com/top-sql-database-guis/

[84] G. Labs, “Grafana: The open observability platform | Grafana Labs,” Grafana Labs. Accessed: Jan. 15, 2024. [Online]. Available: https://grafana.com/

[85] Grafana, “GitHub - grafana/grafana: The open and composable observability and data visualization platform. Visualize metrics, logs, and traces from multiple sources like Prometheus, Loki, Elasticsearch, InfluxDB, Postgres and many more.,” GitHub. Accessed: Jan. 15, 2024. [Online]. Available: https://github.com/grafana/grafana

[86] “Grafana OSS and Enterprise | Grafana documentation,” Grafana Labs. Accessed: Oct. 17, 2023. [Online]. Available: https://grafana.com/docs/grafana/latest/?pg=oss-graf&plcmt=hero-btn-2

[87] Tech Tutorials - David McKone, *How To Install Prometheus And Grafana On Docker*, (Aug. 01, 2023). Accessed: Oct. 17, 2023. [Online Video]. Available: https://www.youtube.com/watch?v=jj38y6f6UpE

[88] DB Tech, *Docker Dashboard Using Grafana, Prometheus & Node Exporter*, (Dec. 21, 2020). Accessed: Oct. 18, 2023. [Online Video]. Available: https://www.youtube.com/watch?v=83LWo7h_hvs

[89] Grafana, “Grafana Play Home - Examples - Dashboards.” Accessed: Oct. 27, 2023. [Online]. Available: https://play.grafana.org/d/000000012/grafana-play-home?orgId=1

[90] MariaDB, “MariaDB Enterprise Open Source Database | MariaDB.” Accessed: Jan. 23, 2024. [Online]. Available: https://mariadb.com/

[91] Microsoft, “Visual Studio Code.” Accessed: Oct. 11, 2023. [Online]. Available: https://code.visualstudio.com/

[92] “Python,” Python Software Foundation . Accessed: Oct. 11, 2023. [Online]. Available: https://www.python.org/

[93] J. VanderPlas, *Python Data Science Handbook: Essential Tools for Working with Data*. O’Reilly Media, 2016.

[94] “draw.io - free flowchart maker and diagrams online.” [Online]. Available: https://app.diagrams.net/

[95] K. Pravossoudovitch, F. Cury, S. G. Young, and A. J. Elliot, "Is red the colour of danger? Testing an implicit red-danger association," *Ergonomics*, vol. 57, no. 4, 2014, doi: 10.1080/00140139.2014.889220.

[96] J. Brooke, "SUS—a quick and dirty usability scale.," 1996.

[97] J. R. Lewis, "The System Usability Scale: Past, Present, and Future," *Int J Hum Comput Interact*, vol. 34, no. 7, 2018, doi: 10.1080/10447318.2018.1455307.

[98] A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the system usability scale," *Int J Hum Comput Interact*, vol. 24, no. 6, 2008, doi: 10.1080/10447310802205776.

[99] J. R. Lewis, J. Brown, and D. K. Mayes, "Psychometric Evaluation of the EMO and the SUS in the Context of a Large-Sample Unmoderated Usability Study," *Int J Hum Comput Interact*, vol. 31, no. 8, 2015, doi: 10.1080/10447318.2015.1064665.

[100] J. R. Lewis and J. Sauro, "Revisiting the Factor Structure of the System Usability Scale," 2017.

[101] J. Sauro and J. R. Lewis, *Quantifying the User Experience: Practical Statistics for User Research*, 2nd ed. 2016.

Anexo A – Proposta de Estágio

Proposta de Estágio

VOID SOFTWARE

Título: Observabilidade em DevSecOps - Visualização de dados

Departamento: DevSecOps

Criador(es): Bernardo Sequeira, Fábio Antunes, Eduardo Piza

Supervisor(es): Bernardo Sequeira

Referência: INTERNSHIP.P4.2023

Contexto

A observabilidade tem vindo a emergir como um conceito crítico no cenário DevSecOps, permitindo que as organizações obtenham conhecimento profundo sobre os seus sistemas de software e processos de segurança [1]. Neste contexto, a visualização de dados na forma de um dashboard desempenha um papel fundamental na tradução de grandes quantidades de dados em conhecimento. Ao combinar observabilidade e visualização de dados, as organizações podem melhorar a sua compreensão do comportamento dos sistemas, detectar anomalias e tomar decisões informadas para garantir que as equipas de operações e de desenvolvimento de software são eficientes dentro do paradigma DevSecOps [2-4].

A observabilidade em DevSecOps vai além das abordagens tradicionais de monitorização, agrega e analisa grandes quantidades de dados de várias fontes, como logs, métricas e eventos de segurança. É aqui que a visualização dos dados, através de um dashboard bem desenhado se torna indispensável. Um dashboard apresenta os dados de observabilidade, visualmente, por meio de gráficos interativos e métricas, fornecendo uma visão geral dos principais indicadores de desempenho, eventos de segurança, métricas de conformidade e outras informações críticas [5].

A visualização de dados é uma “ferramenta” poderosa dentro da ciência de dados que melhora a exploração de dados, análise, desenvolvimento de modelos e comunicação de resultados [6]. Ajuda a compreender os dados, extrair *insights* significativos e transmitir efetivamente as descobertas a vários públicos.

Objetivo

Este estágio foca-se no desenvolvimento de um dashboard que fornece *insights* sobre os resultados de análises e testes de projetos distintos integrados numa pipeline CI/CD. Nomeadamente, visa recolher e estruturar resultados de fontes distintas e apresentá-los num dashboard central. A pipeline CI/CD já existente inclui a execução de ferramentas como Cypress,

SonarQube, WSAP, Dependency-Check, Lizard, SwiftLint, MobSF. Cada ferramenta tem o seu próprio formato de log e relatório, pelo que o desafio começa precisamente na fase de exploração de dados que irá conduzir às fases seguintes num processo de ciência de dados.

Atividades

1. Visão geral das tecnologias e ferramentas utilizadas (CI/CD pipeline)
2. Visão geral do processo de desenvolvimento e padrões de segurança em prática dentro da empresa
3. Realizar o levantamento do estado da arte sobre dashboards e visualização de dados em contextos DevSecOps
4. Identificar as fontes essenciais de dados para o dashboard e desenho o modelo de dados correspondente
5. Desenhar e desenvolver o dashboard de DevSecOps

Referências

[1] Goniwada, S. R., & Goniwada, S. R. (2022). Observability. *Cloud Native Architecture and Design: A Handbook for Modern Day Architecture and Design with Enterprise-Grade Examples*, 661-676.

[2] What is DevOps Observability (Importance & Best Practices) | BrowserStack. (2023, February 10). BrowserStack. https://browserstack.wpengine.com/guide/observability-devops/ [Available online: accessed June 21 2023]

[3] New Relic. (n.d.). 2022 Observability Forecast Spotlight. from https://newrelic.com/sites/default/files/2022-09/new-relic-2022-observability-forecast-infographic.pdf [Available online: accessed June 21 2023]

[4] Jahn, R. (2022, December 6). How to boost SRE productivity with observability-driven DevOps. Dynatrace News. https://www.dynatrace.com/news/blog/getting-started-observability-driven-devops/ [Available online: accessed June 19 2023]

[5] Islam, M., & Jin, S. (2019, November). An overview of data visualization. In *2019 International Conference on Information Science and Communications Technologies (ICISCT)* (pp. 1-7). IEEE.

[6] *Toasa, R., Maximiano, M., Reis, C., & Guevara, D. (2018, June). Data visualization techniques for real-time information—A custom and dynamic dashboard for analyzing surveys' results. In 2018 13th Iberian Conference on Information Systems and Technologies (CISTI) (pp. 1-7). IEEE.*

Versão do Documento

V0.9 - Draft - June 2023 - Author: Eduardo Piza - VOID Labs - VOID Software

Anexo B – "*Security Insights at a Glance: An Integrated Data Approach for Application Security*"

Este anexo apresenta o artigo publicado e apresentado na conferência "*The 18th IEEE International Conference on Application of Information and Communication Technologies (AICT2024)*.

Security Insights at a Glance: An Integrated Data Approach for Application Security

Alexandre Penela
VOID Labs - VOID Software
ESTG - Polytechnic of Leiria
Leiria, Portugal
alexandre.penela@voidsoftware.com

Bernardo Sequeira
VOID Labs - VOID Software
Leiria, Portugal
bernardo.sequeira@voidsoftware.com

Eduardo Piza
VOID Labs - VOID Software
Leiria, Portugal
eduardo.piza@voidsoftware.com

Maria Beatriz Piedade
ESTG - Polytechnic of Leiria
Leiria, Portugal
beatriz.piedade@ipleiria.pt

Rosa Matias
ESTG - Polytechnic of Leiria
Leiria, Portugal
rosa.matias@ipleiria.pt

Catarina I. Reis
VOID Labs - VOID Software
Leiria, Portugal
catarina.reis@voidsoftware.com

***Abstract*—The rise in cybercriminals exploiting web application vulnerabilities in recent years has intensified the need to strengthen security protocols throughout the software development life cycle (SDLC). However, application security visualization remains a largely unexplored field. The current state of the art typically focuses on presenting data from a single type of security tool. This work introduces an innovative prototype solution that allows security experts to visualize aggregated application-related security data from multiple security tools. This integrated and comprehensive view of an application's security posture enables more informed, data-driven decisions regarding security practices during application development.**

***Index Terms*—Analytics, Application Security, Dashboard, Data visualization, Security Tools**

I. Introduction

As the world continues its digital transition, people have become more reliant on web applications to execute essential daily tasks. The simultaneously increase of usage and user numbers in applications translates into a considerable increase in the number of vulnerabilities that, when exploited, expose individuals and organizations [1]. Recently, the rise of Cybercrime has led to huge corporate and institution losses [2].

According to the Data Breach Investigating Report published by Verizon [3], web applications served as the main action vector used by criminals in cybercrime activities. Additionally, new rules and procedures were established within the EU countries to increase data security and privacy standards (e.g. GDPR). The rising concern by companies to meet all the current security and compliance criteria has led to the development of the DevSecOps concept.

DevSecOps is a set of practices whose goal is to integrate security measures in all Software Development Life Cycle (SDLC). To effectively implement these practices, development, security, and operations teams must maintain continuous collaboration [4], [5] (Figure 1).

This research work was supported by *VOID SOFTWARE, S.A.*, to whom the authors gratefully acknowledge the generous support.

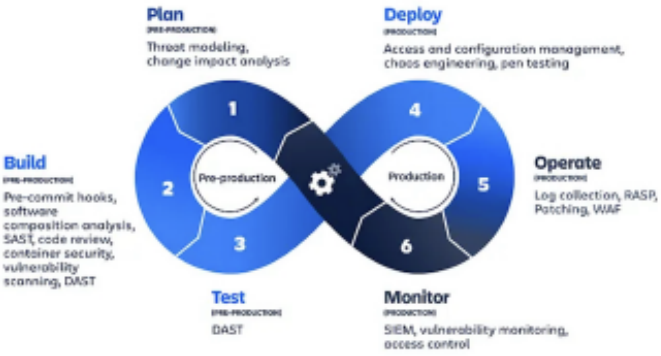


Fig. 1. DevSecOps tools and practices [6].

One of DevSecOps common and essential practices is the implementation of automated security tests through security scanners. The proper adoption of these tools allows the identification of vulnerabilities in the early stages, making it easier and cheaper to fix, in comparison to scenarios where vulnerabilities are discovered in more advanced stages of development [7]. A vast amount of automated analysis security tools are available, with distinct functionalities, providing efficient security checks [8]. These tools have reporting systems but generally fail to visually display their findings, either by not having any visual elements in the report or by possessing only simplistic and limited coloring elements that are insufficient to visualise data adequately. Additionally, most security scanners don't offer options that allow the integration of data between them and don't incorporate a temporal dimension in the report's data analysis. It is still a challenging task to successfully address the application security status and monitor its changes over time.

An effective and integrated analysis of report output data generated by these scanners, utilizing specific graphical displays through an interactive dashboard, would improve the

monitoring and evaluation of both individual and overall security statuses of developed applications. This approach would also enable data-driven decisions when adopting or adjusting security policies within the company [9], thereby strengthening the security layer throughout the application's SDLC.

The aim of this work is to develop a prototype by integrating all relevant data retrieved from the reports of several different types of security tools. Dynamic Application Security Testing (DAST) [10], Static Application Security Testing (SAST) [11] and Software Component Analysis (SCA) [12] will have their results combined into a unique dataset. It will be used to create meaningful visualizations of that data, through a custom designed dashboard with several interactive features such as filters and drill-down options and continuously monitor and evaluate that status over time.

In the next section the related work is reviewed, section III describes the solution architecture and all its development decisions, including the dashboard design. An analysis regarding the solution benefits and issues, as well as a comparative analysis with existing tools. Finally, the last section will address future steps.

II. Literature Review

The literature review was conducted according to the PRISMA statement guide [13]. The search was performed on four different databases : *Google Scholar*, *ACM Digital Library*, *IEEE Xplore* and *Springer*. The search query was defined as "application security" AND vulnerability AND (dashboard or "data visualization"). Results were filtered in order to focus on the most recent work, selecting only those published in the last five years (1 Jan 2019 - present). Records written in a non-English language and duplicates were also excluded. Records were sorted by relevance and the first 20 records from each database were retrieved and grouped in a single list. A total of 45 studies passed to the screening stage. From the studies analysed in the screening stage, by two distinct evaluators, 12 were considered relevant in the context of this work and 2 were excluded as they couldn't be fully accessed. The remaining studies went through a deep evaluation according to the following criteria: scientific quality, research scope, writing quality and relevance. This evaluation resulted in the exclusion of 3 more articles. In addition to the regular procedure, PRISMA incorporates a concept to include other records that were deemed relevant for the literature review but were not originally identified through the regular research methodology. Three more relevant publications were added. A total of 10 records were included in the literature review.

Several studies have been conducted within the field of security and cybersecurity visualization. [14] defined 11 focus areas to address organization's cybersecurity maturity level, with application security being one of them. Within security visualization a study found that although many areas of security were addressed in the area, specific studies regarding web application security and vulnerabilities were limited [15].

The studies conducted in application the security visualization area mainly focused in SAST to visualize software vulnerabilities. [16] used a treemap to visually display software vulnerabilities identified by the FindBugs scanner. [17] developed a software vulnerability visualization tool using a user-centered approach at the design phase. Android applications were loaded into the tool and vulnerabilities information identified in static analysis were shown in several visualizations. Visualization tool created in [18] combined data from multiple SAST reports with provenance information related to application's source code. The integration and storage of provenance information within the data model enables the filtering of events in software development and perform visual analysis of data at the specific times when those events occurred.

Two studies within the scope of application security visualizations used DAST. [19] proposed framework used not only scan results from multiple DAST but also parent-child relationship information of tested URL domains and their respective pages. [8] developed HWAS-V tool in a holistic approach that visualizes outputs of DAST scan results combined with project-level and application-level attributes such as application size, number of modules, number of external libraries. Furthermore, it incorporates approximately 50 different metrics and measures in the Dashboard views. Finally, information of OWASP, WASC and CWE standards is stored in the proposed data model to provide the functionality of associating scan rules to those standards.

Despite some of the mentioned approaches used the results of more than one security scanner as data sources, even integrating data with other application related sources, the outputs only present data from one type of tool (either SAST or DAST). Therefore, to the best of our knowledge, our proposal of integrated approach of using combined data from both SAST and DAST scan results as well as using data related to vulnerabilities from third party libraries from SCA and data from end-to-end (E2E) automation test results it's the first of its kind.

III. Architecture

The proposed solution was designed primarily for Security Experts and for Quality Assurance (QA) Engineers. It proposes an holistic overview allowing some uncovered vulnerabilities to be promptly detected at early stages of development. Thus, through the analysis of the results from an automated End-2-End (E2E) testing tool, the information presented will be more accurate and contextualized. Security Analysts and Information Security Managers will check and analyse the vulnerability information of the applications. QA Engineers will check and analyse the information of the automatic tests.

The proposed solution has a set of functional requirements:

- overview of all the vulnerabilities of a project;
- overview of the total number of vulnerabilities identified in a project over time;
- search and filter vulnerabilities by security tool, severity class and date;

- access vulnerability details on demand;
- view and analyse results from Cypress automated tests of a project.

It should also comply to most of the most common non functional requirements that are mandatory for these tools, such as performance, scalability, flexibility, availability and accessibility.

As mentioned earlier, the proposed solution will use the security scanner's reports as data sources as well as vulnerability data from third party libraries and E2E automation tests results. There was an initial observation and understanding of the data shown in the scanning reports. Following this, an Extract, Transform, Load (ETL) process was performed using an automated Python script to process the relevant data. A relational data model was developed and implemented in MariaDB to store the data. Subsequently, Grafana [20] was used as a visualization tool to design a dashboard, containing several data visualization widgets that will be used to present useful insights. The expected result is an integrated solution with capabilities that allow a proper analysis and monitoring of vulnerabilities detected in web applications developed by the company and evaluate its security status. For the development of the proof of concept, data from three different application projects were used, anonimized due confidentiality matters. Figure 2 presents a general overview of the architecture.

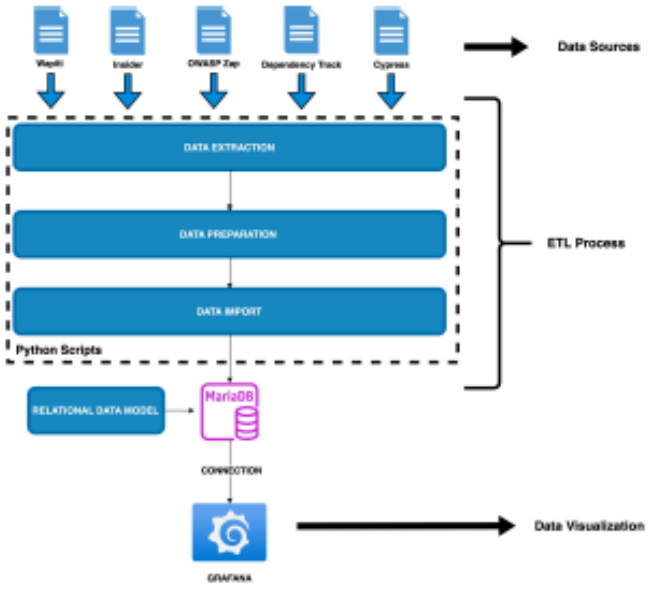


Fig. 2. Architecture of the solution developed.

A. Data sources

Two security tools and one End-2-End (E2E) automation tool were used as data sources: WASP [21], Dependency Track [22] and Cypress [23]. WSAP is composed by three distinct application security scanners: Insider CLI [24], Wapiti [25] and OWASP ZAP [26]. Dependency Track was the SCA tool used to retrieve data from third party dependencies vulnerabilities. Cypress is distinct from the other two tools, it runs automated system tests on Web applications. These test results can still be related to application security as malfunctions in app developed features can lead to security issues e.g. improper privileged access to information or user identification and authentication failure.

B. Data Extraction and Transformation (ETL)

Python scripts were developed to automatically access and extract data from WSAP scan results, Dependency Track third-party dependency vulnerabilities present in the Software Bill of Materials (SBOM) report, and Cypress test results.

Although these tools run independently from each other, their structure and actions on data are very similar. An HTTP request grants access to the security tool report's data, allowing its extraction. For each tool, a dataframe was created, and at the data cleaning stage, attributes were analyzed and selected by relevance in collaboration with the organization's information security manager. Irrelevant variables were excluded from the dataframe and null values were either removed or replaced with "NA" string values, depending on the type of information provided by the attribute.

At the processing phase, the structure of the DataFrames was normalized to match with the schema of the data model. New attributes were also created such as a date attribute that corresponds to the creation date of the scan/test reports. It is essential as it incorporates a temporal dimension into the security analysis of the applications.

C. Relational Data Model

Different tools produce distinct data attributes. In our work, due to the significant differences between the tools and its attributes, the choice to simplify the data to only accommodate the common attributes to all the tools was not a valid solution, as it would result in a substantial relevant information loss. Thus, decisions regarding the relational model development had into account model flexibility and scalability criteria i.e. the model should be flexible enough to integrate the diverse types of attributes and allow incorporating other security tools and possibly other security-related data sources in the model without the need to increment the table numbers in the model. The final version of the relational data model is presented in Figure 3.

WSAP and Dependency Track related data is stored *Tool_table*, *Attribute_table*, *Vulnerability_table* and the main table *Values_table*, which stores the vulnerability attribute values. Other tables store information of the security tools, the vulnerability and the extracted attributes. A dedicated table stores project information (Project ID and Project Name) and attributes from Cypress automatic test results were stored in 3 new tables, *Cypress_Features_table*, *Cypress_Scenarios_table*, *Cypress_Steps_table*.

MariaDB was used to implement the relational data model. Primary and foreign keys were defined to establish inter-table relationships. For each column, a data type and specific constraints were set to ensure data integrity and reliability. After the tables were created, a script was executed to populate the tables with data.

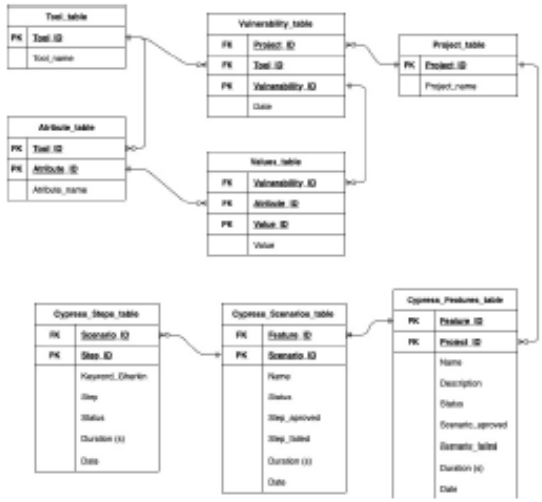


Fig. 3. Relational Data Model.

IV. Dashboard Design

Grafana was the selected tool to create the Dashboard. A MySQL plugin was used to connect the database to Grafana. The dashboard consists of three main sections and overall filters:

A. Section Vulnerabilities : Overall

Provides a general overview of the vulnerabilities identified by security scanners in the project (Figure 4). It comprises three distinct visualization panels. From left to right, the first visualization is a card displaying the total number of vulnerabilities currently found in the application. The central panel is a bar graph that groups the vulnerabilities by their severity classes. Colors used are associated with the severity class of vulnerabilities, promoting visual analysis. Values are sorted by descending order, so that the class with the highest number of vulnerabilities is on the left and the class with the lowest number is on the right. The rightmost panel is a line graph that shows the number of vulnerabilities detected in each security scanner over time.

B. Section Vulnerabilities : Detailed Info

This section contains the "on-demand" details of the vulnerabilities, presented in a listing. The build-in filters enhance the data accessibility, as users can effectively search specific variables or quickly locate certain data points, either through text searches or by selecting values from a list. This functionality enables users to focus their analysis on subsets of vulnerabilities.

Fig. 4. Overview of the vulnerabilities overall information section.

C. Section Cypress

This section holds 4 panels. Two pie charts show the number and proportion of features and scenarios that passed or failed in comparison to the total features and scenarios. An additional panel consists of a line graph showing the number of features tested by Cypress, over time.

The last panel is a table listing the names of the features tested and the total duration of the automated tests for each feature. A measurement bar is associated with the test duration, serving as a visual indicator, i.e. the longer the bar, the longer the testing duration for the given feature. This bar uses a gradual color pattern, with shades of green representing shorter duration and shades of red indicating longer test duration. The total duration of tests for all features is also displayed (Figure 5).

Fig. 5. Cypress tests information section.

D. Filters

Users are able to filter vulnerability data by security tool, by the severity classes and by the date of the security report, and even cross-filter the results (Figure 6). The first 2 filters have the particularity of multiple selection, i.e. they allow the selection of more than one tool or severity class. The date serves as a temporal filter and uses time intervals predefined by Grafana, however it is possible for the user to set a specific date interval to filter the information.

Users can use filters simultaneously (cross-filter), thus allowing data to be analyzed in a more flexible, specific and in-depth way.

V. Discussion

We created a prototype solution, using an integrated approach, to visualize relevant data from multiple types of security tools reports in a customised Dashboard. Despite our solution has not yet been evaluated by users, the dashboard was designed considering preliminary feedback given by the Information Security Manager and a QA tester.

A. Solution Analysis

The designed solution meets the requirements as it allows the data from security scanners to be properly processed, transforming them into relevant insights that will serve as a basis for future decision-making regarding application security protocols and policies. The solution's flexibility and scalability assures that new attributes and new security scanners can be easily added and incorporated in the current database.

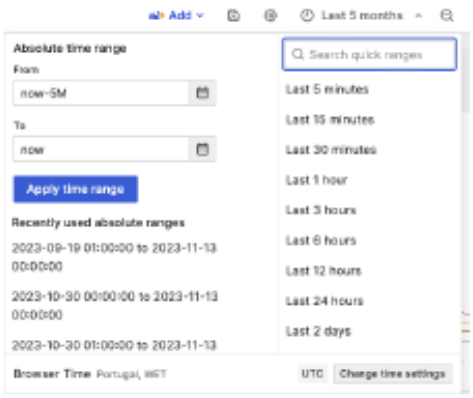


Fig. 6. Date time filter.

The degree of complexity of the solution is not excessively high, making it easier to implement and maintain. The scripts developed to perform the necessary tasks in a automated way guarantee high performance, greatly reducing human error, with faster execution time compared to an architecture more dependent on manual processes.

The Dashboard provides an holistic, integrated and detailed view of the security status of applications, aligned with the company's objectives and needs. Each dashboard section is logically and clearly organized, presenting information succinctly and cohesively to facilitate data analysis and understanding. Dynamic filters facilitate data analysis and interpretation by enabling the users to explore data interactively, thus increasing the performance and utility of the dashboard. The temporal filter is particularly useful in identifying trends, comparing the state of the application in different time periods or analyzing the entire history of vulnerabilities.

Regardless, our solution presents some constraints. For instance, our solution focus mainly on statistical visualizations of data, lacking other types of metrics for comprehensive data analysis. Another issue identified is the inability to separate and filter vulnerabilities by specific development events. If two builds occur on the same day, it is not possible to associate vulnerabilities found with their respective builds. This limitation hampers the ability to precisely trace vulnerabilities to particular development activities.

B. Comparative Analysis

[16] has filter capabilities based on severity, type, and location within the code as well as a details on demand pane. However, the visualization is a treemap based only which is not suitable for all types of data analysis and limit the tool's effectiveness in presenting different aspects of the data. It also fails to use colors to represent data, and does not display the absolute number of vulnerabilities.

[17] provides a in-depth visualization of a single software vulnerability report. The tool is compatible with different scanners. Its main features include overall vulnerabilities view, search and filter options, details on demand section and a comparative view between different versions of the application. Three main limitation were identified: vulnerabilities matching problems in report comparison view, inadequate color association to the data and the inability to apply multiple filters simultaneously.

[18] combines information for SAST scanners with provenance information from the codebase. This approach allows users to analyse software development events of interest in an isolated manner. The interface functionalities include zooming, data comparison, selection and cross-filtering with three levels of data granularity, thus providing an interactive and in-depth analysis. Tool limitation resides on the fact that information is grouped by CWE category and this attribute is not accessible for all warnings, resulting in an incomplete visualization of warnings. It also lacks a search function.

[19] focuses on vulnerabilities identified by DAST scanners using website hierarchical structure along with website vulnerability scan results, to provide statistical visualizations of the information. Its interactive features include node collapsing/expanding, filtering, zooming, and details-on-demand visualization. Limitations are related to the visualization technique implemented, with applications with more than 70 nodes posing an analytical challenge for the users.

[8] presents data from several DAST scanner outputs. A strong focal point of this tool is the definition and implementation of about 50 metrics in dashboard visualizations to address application's security status.

Summarily, every tool has its own strong points and constrains. Nevertheless, none of the previously discussed tools integrate data from multiple types of security tools into a unified dashboard. Moreover, some of the visualization tools don't consider or at least don't mention a temporal dimension in vulnerability analysis, thus not making them properly suited to monitor and valuate application security over time.

VI. Conclusion

This work focused on visualizing application related security data using reports from different types of security tools through an integrated and unified approach. The solution developed, meets the requirements initially established, offering a comprehensive and holistic view of the application's security status.

Despite being a proof of concept, comparative analysis demonstrates that our solution encompasses most, if not more, of the functionalities present in other similar tools, which focus on specific types of vulnerabilities or don't include a time dimensional axis in data analysis. Therefore, according to our research and best knowledge, the tool developed constitutes the first of its kind. This approach not only enhances the ability to monitor and evaluate security status comprehensively but also sets a new standard for future tools in application security visualization domain.

We have identified the need for a formal an biased-free evaluation, ensuring that the tool meets the needs of all user types. This evaluation will be conducted using a System Usability Scale (SUS) questionnaire, which will be completed

by users from both Security and QA teams. Tool improvement will be done through an interactive process, based on user feedback.

Additionally, future steps include addressing the constraints already identified and described. Taking advantage of our data model, new types of application security related data will be added. Build related information will be included and stored in the database, so vulnerabilities can be associated with specific application development events. Finally, potential metrics will be listed and analysed to include in the dashboard.

References

[1] M. Alenezi and Y. Javed, "Open source web application security: A static analysis approach," in *2016 International Conference on Engineering & MIS (ICEMIS)*, 2016. DOI: 10.1109/ICEMIS.2016.7745369.

[2] M. Wilczek. "Average cost of a data breach in 2020: $3.86m." (2020), [Online]. Available: https://www.darkreading.com/vulnerabilities-threats/average-cost-of-a-data-breach-in-2020-3-86m (visited on 04/30/2024).

[3] P. Langlois, A. Pinto, D. Hylender, and S. Widup, *Dbir 2023 data breach investigations report 10k 20k 30k about the cover*, Jun. 2023. DOI: 10.13140/RG.2.2.32362.70085.

[4] H. Myrbakken and R. Colomo-Palacios, "Devsecops: A multivocal literature review," Sep. 2017, pp. 17–29. DOI: 10.1007/978-3-319-67383-7_2.

[5] T. Rangnau, R. Buijtenen, F. Fransen, and F. Turkmen, "Continuous security testing: A case study on integrating dynamic security testing tools in ci/cd pipelines," Oct. 2020, pp. 145–154. DOI: 10.1109/EDOC49727.2020.00026.

[6] "DevSecOps Definition, Best Practices and Tools." (Jun. 2023), [Online]. Available: https://medium.com/@cloud_tips/devsecops-definition-best-practices-and-tools-1789587d165a.

[7] M. Alenezi and Y. Javed, "Open source web application security: A static analysis approach," in *2016 International Conference on Engineering & MIS (ICEMIS)*, 2016, pp. 1–5. DOI: 10.1109/ICEMIS.2016.7745369.

[8] F. Özdemir Sönmez and B. Gunel, "Holistic web application security visualization for multi-project and multi-phase dynamic application security test results," *IEEE Access*, vol. PP, pp. 1–27, Feb. 2021. DOI: 10.1109/ACCESS.2021.3057044.

[9] S. H. et al., "A new approach to data analysis using machine learning for cybersecurity," *Big Data and Cognitive Computing*, vol. 7, no. 4, 2023. [Online]. Available: https://www.mdpi.com/2504-2289/7/4/176.

[10] "Sast vs. dast: What's the best method for application security testing?" (2024), [Online]. Available: https://www.synopsys.com/blogs/software-security/sast-vs-dast-difference.html (visited on 01/18/2024).

[11] P. J. Yang J. Tan L. and D. K.A, "Towards better utilizing static application security testing," *IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice*, 2019.

[12] "Software composition analysis: Key to app security & compliance." (2024), [Online]. Available: https://www.zimperium.com/glossary/software-composition-analysis-sca (visited on 01/18/2024).

[13] M. J. P. et al., "The prisma 2020 statement: An updated guideline for reporting systematic reviews," *BMJ*, vol. 372, n71, Mar. 2021. DOI: 10.1136/bmj.n71.

[14] B. Yiğit Özkan, S. Lingen, and M. Spruit, "The cybersecurity focus area maturity (cysfam) model," *Journal of Cybersecurity and Privacy*, vol. 1, pp. 119–139, Feb. 2021. DOI: 10.3390/jcp1010007.

[15] F. Özdemir Sönmez, "Security visualization extended review:issues, classifications, validation methods, trends, extensions," in May 2018, pp. 152–197.

[16] H. Assal, S. Chiasson, and R. Biddle, "Cesar: Visual representation of source code vulnerabilities," Oct. 2016, pp. 1–8. DOI: 10.1109/VIZSEC.2016.7739576.

[17] S. L. Reynolds, T. Mertz, S. Arzt, and J. Kohlhammer, "User-centered design of visualizations for software vulnerability reports," *2021 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pp. 68–78, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:244957211.

[18] A. Schreiber, T. Sonnekalb, and L. Kurnatowski, "Towards visual analytics dashboards for provenance-driven static application security testing," Oct. 2021, pp. 42–46. DOI: 10.1109/VizSec53666.2021.00010.

[19] T. T. Dang and T. Dang, "An extensible framework for web application vulnerabilities visualization and analysis," in Jan. 2014, pp. 86–96, ISBN: 978-3-319-12777-4. DOI: 10.1007/978-3-319-12778-1_7.

[20] *Grafana: The open observability platform | Grafana Labs*, [Online; accessed 24. Jan. 2024], Jan. 2024. [Online]. Available: https://grafana.com.

[21] J. P. B. F. C. Marques, "Web security application project," unpublished, 2021.

[22] *OWASP Dependency-Track | OWASP Foundation*, [Online; accessed 24. Jan. 2024], Nov. 2023. [Online]. Available: https://owasp.org/www-project-dependency-track.

[23] *Testing Frameworks for Javascript | Write, Run, Debug | Cypress*, [Online; accessed 24. Jan. 2024], Jan. 2024. [Online]. Available: https://www.cypress.io.

[24] insidersec. "Insider." (2024), [Online]. Available: https://github.com/insidersec/insider (visited on 01/18/2024).

[25] "Wapiti : A free and open-source web-application vulnerability scanner in python." (2023), [Online]. Available: https://wapiti-scanner.github.io (visited on 01/18/2024).

[26] "Owasp zed attack proxy project." (2024), [Online]. Available: https://www.zaproxy.org (visited on 01/18/2024).

Anexo C – Folha de cálculo da fase de *screening* do PRISMA

ID	Tittle	Abstract	Avaliação	Avaliação Catarina	Avaliação Total
1	Holistic Web Application Security Visualization for Multi-Project and Multi-Phase Dynamic Application Security Test Results	As the number of web applications and the corresponding number and sophistication of the threats increases, creating new tools that are efficient and accessible becomes essential. Although there is much research concentrating on network security visualizations, there are only a few studies considering the web application vulnerabilities' possible visualization options. Consequently, to fill this gap, this research centers around a novel perception configuration to improve web application vulnerability monitoring. This study forms a generic data structure based on data sources that might be readily associated and commonly available for the majority of the web applications. The primary contribution of this study is a new dashboard tool for visualizing dynamic application security test results. Another contribution is the metrics/ measures that the tool presents. The paper also describes a validation study in which participants answered quiz questions upon using the tool prototype. For the case study, sample data has been generated using the OWASP ZAP scanner tool and a prototype has been implemented to be used for validation purposes. This study allows the investigation of fifty metrics/measures for the multi-project/phase environment that enhances its benefits if the user aims to monitor a series of analyses' results and the changes between them for more than one web project.	2	2	4
2	Towards Visual Analytics Dashboards for Provenance-driven Static Application Security Testing	The use of static code analysis tools for security audits can be time consuming, as the many existing tools focus on different aspects and therefore development teams often use several of these tools to keep code quality high and prevent security issues. Displaying the results of multiple tools, such as code smells and security warnings, in a unified interface can help developers get a better overview and prioritize upcoming work. We present visualizations and a dashboard that interactively display results from static code analysis for “interesting” commits during development. With this, we aim to provide an effective visual analytics tool for code security analysis results.	2	2	4
3	Security visualization infrastructures, techniques, and methodologies for improved enterprise security	This thesis focuses on providing designs to allow monitoring of the security status of enterprises at the organization level. The audience of this research is all enterprise level IT and security experts, and the other users who may be engaged in the security visualization designs, including the top level management. Numerous tools and programs are being used in organizations to analyze and overcome security vulnerabilities. However, the outputs of these programs are rarely understood clearly. During this study, existing security visualization requirements and designs along with the corresponding technologies used for security visualization were examined. For the sake of being user- centric, a visualization requirements survey was held. The results of the literature review and the survey were converted to a substantial requirement set for a generic enterprise security visualization infrastructure. This infrastructure was then implemented using industry’s best standards and the contemporary big data solutions. The resulting design was validated through the use of expert reviews. Later, one of the favorite security visualization subjects for the enterprises, namely web application security was handled. A dashboard type holistic design to visualize black-box vulnerability test results was proposed along with forty plus metrics and measures. SIEM systems were also examined for their custom data visualization capabilities in parallel to this part of the study. Finally, security management related issues for the organizations was focused. In this part of the study, a decision support system for the optimization of security costs which relies on analytical methods and uses treemap type visualizations to visualize the threats, risks, corresponding precautions, and the costs was proposed. A real-world case study was used to demonstrate the benefits of this system.	2	2	4

Figura 32 – Parte da folha de cálculo utilizada na fase de screening do PRISMA

Anexo D – Scripts de extração de dados

Este anexo apresenta os *scripts* completos desenvolvidos para a extração e transformação de dados a partir dos relatórios gerados pelas ferramentas de segurança. Importa relembrar que os *scripts* utilizados nos 3 projetos são idênticos, pelo que a apresentação de um *script* de cada ferramenta de um único projeto é representativa dos *scripts* para os restantes projetos.

```
 1. import json
 2. import pandas as pd
 3. with open("/Users/alexandre.penela/Documents/Report
Extration/WSAP/Project_A/wapiti_project_A.json", "r") as projeto_nome:
 4.     data= json.load(projeto_nome)
 5.
 6. #Lista de variáveis
 7.
 8. Vulnerabidade = []
 9. Vulnerability_type = []
10. Info = []
11. curl_command = []
12. http_request = []
13. level = []
14. method = []
15. wstg = []
16.
17. #Loop de extração
18.
19. for key in data:
20.     for vul in data[key]:
21.         Vulnerabidade.append(key)
22.         curl=vul.get('curl_command', None)
23.         http=vul.get('http_request', None)
24.         inf=vul.get('info', None)
25.         vul_type=vul.get('vulnerability_type', None)
26.         lev=vul.get('level', None)
27.         met=vul.get('method', None)
28.         wst=vul.get('wstg', None)
29.         curl_command.append(curl)
30.         http_request.append(http)
31.         Info.append(inf)
32.         Vulnerability_type.append(vul_type)
33.         level.append(lev)
34.         method.append(met)
35.         wstg.append(wst)
36.
37. #Tabela
38. dwap= {'VULNERABILITY_CLASS': Vulnerabidade,'VULNERABILITY_TYPE':Vulnerability_type,
'INFO':Info,  'CURL COMMAND TO REPLICATE': curl_command, 'HTTP_REQUEST': http_request, '
SEVERITY_LEVEL':level, 'METHOD':method, 'WSTG':wstg}

40. df=pd.DataFrame(dwap)
41. df=df.replace('\n', '',regex=True)
42.
43. #Exportar output para csv
44. ficheiro_csv="/Users/alexandre.penela/Documents/Report
Extration/WSAP/Project_A/wapiti_extract_Project_A.csv"
45. df.to_csv(ficheiro_csv, index=False)
```

Listagem 14 – *Script* completo de extração de dados do Wapiti

```
 1. import json
 2. import pandas as pd
 3.
 4. with open("/Users/alexandre.penela/Documents/Report
Extration/WSAP/Project_A/insider_project_A.json", "r") as projeto_nome:
 5.     data= json.load(projeto_nome)
 6.
 7. #Lista de variáveis
 8.
 9. Vulnerabilidade= []
10. Classe=[]
11. ClassM=[]
12. Column=[]
13. cvss=[]
14. cwe=[]
15. Description=[]
16. Line=[]
17. Method=[]
18. Recomendation=[]
19. Vul_id=[]
20.
21. #Loop de extração
22.
23. for key in data:
24.     for vul in data[key]:
25.         Vulnerabilidade.append(key)
26.         cl=vul.get('class', None)
27.         clm=vul.get('classMessage', None)
28.         col=vul.get('column', None)
29.         css=vul.get('cvss', None)
30.         cw=vul.get('cwe', None)
31.         des=vul.get('description', None)
32.         l=vul.get('line', None)
33.         met=vul.get('method', None)
34.         re=vul.get('recomendation', None)
35.         vu=vul.get('vul_id', None)
36.         Classe.append(cl)
37.         ClassM.append(clm)
38.         Column.append(col)
39.         cvss.append(css)
40.         cwe.append(cw)
41.         Description.append(des)
42.         Line.append(l)
43.         Method.append(met)
44.         Recomendation.append(re)
45.         Vul_id.append(vu)
46.
47. #Tabela
48.
49. dins= {'VULNERABILITY_CLASS': Vulnerabilidade,'DESCRIPTION': Description, 'FILE': Classe,
'CLASS_MESSAGE': ClassM, 'CVSS': cvss, 'CWE': cwe,'METHOD': Method, 'RECOMENDATION':
Recomendation}
50.
51. df=pd.DataFrame(dins)
52. df=df.replace('\n', '',regex=True)
53.
54. #Exportar output para csv
55. ficheiro_csv="/Users/alexandre.penela/Documents/Report
Extration/WSAP/Project_A/insider_extract_Project_A.csv"
56. df.to_csv(ficheiro_csv, index=False)
```

Listagem 15 – *Script* completo de extração de dados do Insider

```
import json
import pandas as pd

with open("/Users/alexandre.penela/Documents/Report Extration/WSAP/Project_A/zap_project_A.json", "r") as projeto_nome:
    data= json.load(projeto_nome)

#Lista de variáveis

Vulnerabilidade= []
Alert=[]
AlertRef=[]
Attack=[]
Confidence=[]
cweid=[]
Description=[]
Evidence=[]
Id=[]
MessageId=[]
Method=[]
Name=[]
Other=[]
Param=[]
PluginId=[]
Reference=[]
Risk=[]
Solution=[]
Sourceid=[]
Tags=[]
Url=[]
WASC_ID=[]

#Loop de extração

for key in data:
    for vul in data[key]:
        Vulnerabilidade.append(key)
        al=vul.get('alert', None)
        ar=vul.get('alertRef', None)
        att=vul.get('attack', None)
        con=vul.get('confidence', None)
        cwe=vul.get('cweid', None)
        des=vul.get('description', None)
        ev=vul.get('evidence', None)
        i=vul.get('id', None)
        mi=vul.get('messageId', None)
        me=vul.get('method', None)
        na=vul.get('name', None)
        o=vul.get('other', None)
        par=vul.get('param', None)
        pid=vul.get('pluginId', None)
        re=vul.get('reference', None)
        ri=vul.get('risk', None)
        so=vul.get('solution', None)
        si=vul.get('sourceid',None)
        t=vul.get('tags', None)
        u=vul.get('url', None)
        w=vul.get('wascid', None)
        Alert.append(al)
        AlertRef.append(ar)
        Attack.append(att)
        Confidence.append(con)
        cweid.append(cwe)
        Description.append(des)
        Evidence.append(ev)
        Id.append(i)
        MessageId.append(mi)
        Method.append(me)
        Name.append(na)
```

```
69.         Other.append(o)
70.         Param.append(par)
71.         PluginId.append(pid)
72.         Reference.append(re)
73.         Risk.append(ri)
74.         Solution.append(so)
75.         Sourceid.append(si)
76.         Tags.append(t)
77.         Url.append(u)
78.         WASC_ID.append(w)
79.
80. #Tabela
81.
82. dzap= {'VULNERABILITY_CLASS': Vulnerabilidade, 'ALERT': Alert, 'ALERT_REFERENCE':
AlertRef, 'DESCRIPTION':Description,'CONFIDENCE':Confidence, 'CWE_ID':cweid,
'EVIDENCE':Evidence, 'MESSAGE_ID': MessageId, 'NAME':Name, 'PARAM':Param, '
PLUGIN_REF':PluginId,'REFERENCE': Reference,'SOLUTION':Solution, 'TAGS':Tags, ' URL':Url,
'WASC_REF':WASC_ID}
83.
84. df=pd.DataFrame(dzap)
85. df=df.replace('\n', '',regex=True)
86.
87. #Exportar output para csv
88. ficheiro_csv="/Users/alexandre.penela/Documents/Report
Extration/WSAP/Project_A/zap_extract_Project_A.csv"
89. df.to_csv(ficheiro_csv, index=False)
```

Listagem 16 – *Script* completo de extração de dados do OWASP Zap

```
 1. import json
 2. import pandas as pd
 3. from datetime import date
 4.
 5. projeto_nome="/Users/alexandre.penela/Documents/Report
Extration/cypress/Project_A/Project_A_report.json"
 6.
 7. f=open(projeto_nome)
 8. dados=json.load(f)
 9.
10. #atributos da tabela
11.
12. Nome_feat=[]
13. desc_feat=[]
14. falhas=[]
15. aprovados=[]
16. estado=[]
17. Duracao=[]
18.
19. for i in dados :
20.     i= str(i)
21.     des=i.find("'description':") #descrição
22.     des_fin=i.find("elements")
23.     desc=i[des+20:des_fin-4]
24.     descript=desc[0:1023]
25.     desc_feat.append(descript)
26.     id=i.find("'id'")   #Nome
27.     id_fin=i.find(';',id)
28.     feat=i[id+5:id_fin].replace('-',' ').replace("'","")
29.     Nome_feat.append(feat)
30.
31.     volta=i.count("Scenario") #Cenários aprovados e falhados
32.     pa=int(volta)
33.     scene=i.find("Scenario")
34.     scene_fin=i.find("Scenario",scene+1)
35.     fail=i.count("failed",scene,scene_fin)
```

```
    if fail > 0:
        pa=pa-1
    else:
        pa=pa
    volta= volta -1
    while volta > 0:
        scene=i.find("Scenario",scene_fin)
        scene_fin=i.find("Scenario",scene+1)
        fail=i.count("failed",scene,scene_fin)
        if fail > 0:
            pa=pa-1
        else:
            pa=pa
        volta= volta -1
    aprovados.append(pa)
    falha=int(i.count("Scenario"))-int(pa)
    falhas.append(falha)
    if falha > 0: #Coluna Estado
        est = "Failed"
    else:
        est= "Passed"
    estado.append(est)

#Função de extração de duração
def soma_duracao(dici):
    total=0
    for elemento in dici['elements']:
        for passo in elemento['steps']:
            total += passo['result']['duration']
    return total

#Atributo duração
for i in dados:
    soma_duracao(i)
    soma_total=soma_duracao(i)/1000000000
    Duracao.append(round(soma_total))

#Tabela
cypress_feat= {'Name': Nome_feat, 'Description': desc_feat, 'Status': estado,
'scenario_aproved':aprovados, 'Scenario_failed':falhas, 'Duration (s)':Duracao}

df=pd.DataFrame(cypress_feat)

#Atributo data

data= date.today()
df['Date']= data
df["Date"]= pd.to_datetime(df["Date"], dayfirst= True)
df["Date"]=df["Date"]

#Exportar output para csv
ficheiro_csv="/Users/alexandre.penela/Documents/Report
Extration/cypress/Project_A/features/tabela_features.csv"
df.to_csv(ficheiro_csv, index=False)
```

Listagem 17 – *Script* completo de extração de dados das *features* do Cypress

```
import json
import pandas as pd
from datetime import date

projeto_nome="/Users/alexandre.penela/Documents/Report
Extration/cypress/Project_A/Project_A_report.json"

f=open(projeto_nome)
```

```
dados=json.load(f)

#Função de extração de dados

feature_id=[]
cenario=[]
step_aproved= []
step_failed = []
estado=[]
tempo=[]

feature=1

def dados_extrat(dados):

    for elemento in dados['elements']:
        cen=elemento['name']
        cenario.append(cen)
        passed=0
        failed=0
        dur=0
        for passo in elemento['steps']:
            dur += passo['result']['duration']
            if passo['result']['status'] == "passed":
                passed +=1
            else:
                failed += 1
        if failed > 0 :
            est= "failed"
        else:
            est= "passed"
        total=int(dur)/1000000000
        tempo.append(round(total,ndigits=1))
        step_aproved.append(passed)
        step_failed.append(failed)
        estado.append(est)
        feature_id.append(feature)

#Extração de dados dos cenários em cada feature

for i in dados:
    dados_extrat(i)
    feature=feature +1


#Tabela
cypress_cena= {'Feature_ID': feature_id, 'Name': cenario,  'Status':estado,
'Steps_aproved':step_aproved, 'Steps_failed': step_failed, 'Duração (s)':tempo}

df=pd.DataFrame(cypress_cena)

##Variável data

data= date.today()
df['Date']= data
df["Date"]= pd.to_datetime(df["Date"], dayfirst= True)
df["Date"]=df["Date"]

#Exportar output para csv
ficheiro_csv="/Users/alexandre.penela/Documents/Report
Extration/cypress/Project_A/cenários/tabela_cenarios.csv"
df.to_csv(ficheiro_csv, index=False)
```

Listagem 18 – *Script* completo de extração de dados dos cenários do Cypress

```
 1. import json
 2. import pandas as pd
 3. from datetime import date
 4.
 5. projeto_nome="/Users/alexandre.penela/Documents/Report
Extration/cypress/Project_A/Project_A_report.json"
 6.
 7. f=open(projeto_nome)
 8. dados=json.load(f)
 9.
10. #Função de extração de dados
11.
12. Scenario_ID = []
13. keyword_G=[]
14. estado=[]
15. tempo=[]
16. etapa=[]
17.
18. Scenario= 1
19. def dados_extrat(dados):
20.
21.     for elemento in dados['elements']:
22.         global Scenario
23.         for passo in elemento['steps']:
24.             et=passo['keyword']
25.             et_desc=passo.get('name', None)
26.             etapa.append(et_desc)
27.             keyword_G.append(et)
28.             est=passo['result']['status']
29.             dur= passo['result']['duration']
30.             estado.append(est)
31.             tempo.append(dur/1000000000)
32.             Scenario_ID.append(Scenario)
33.         Scenario += 1
34.
35.
36.


37. #Extração de dados dos steps de cada cenário em cada feature
38.
39. for i in dados:
40.     dados_extrat(i)
41.
42.
43. #Tabela
44. cypress_cena= {'Scenario_ID': Scenario_ID, 'Keyword Gherkin': keyword_G, 'Step': etapa,
'Status':estado, 'Duration (s)':tempo}
45.
46. df=pd.DataFrame(cypress_cena)
47.
48. #Variável data
49.
50. data= date.today()
51. df['Date']= data
52. df["Date"]= pd.to_datetime(df["Date"], dayfirst= True)
53. df["Date"]=df["Date"]
54.
55. #Exportar output para csv
56. ficheiro_csv="/Users/alexandre.penela/Documents/Report
Extration/cypress/Project_A/steps/tabela_steps.csv"
57. df.to_csv(ficheiro_csv, index=False)
```

Listagem 19 – *Script* completo de extração de dados dos *steps* do Cypress

```
 1. import json
 2. import pandas as pd
 3.
 4. ficheiro= open("/Users/alexandre.penela/Documents/Report Extration/Dependency
Track/Project_A/Project_A_api.json")
 5.
 6. dados=json.load(ficheiro)
 7.
 8. ref=[]
 9. id=[]
10. source_name=[]
11. souce_url=[]
12. score_CVSS=[]
13. severity_CVSS=[]
14. method=[]
15. vector=[]
16. description=[]
17.
18. for vul in dados ['vulnerabilities']:
19.     ref.append(vul['bom-ref'])
20.     cve=vul['id']
21.     id.append(cve)
22.     source_name.append(vul['source']['name'])
23.     url_= vul['source']['url']
24.     url_source=url_ + 'vuln/detail/' + str(cve)
25.     souce_url.append(url_source)
26.     des=vul['description']
27.     description.append(des[0:1023])
28.     has_cvssv3 = False
29.     has_cvssv2 = False
30.     for rating in vul ['ratings']:
31.         if rating['method'] == 'CVSSv3':
32.             has_cvssv3 = True
33.             score_CVSS.append(rating['score'])
34.             severity_CVSS.append(rating['severity'])
35.             method.append(rating['method'])
36.             vector.append(rating['vector'])
37.             break
38.
39.     if not has_cvssv3:
40.         for rating in vul['ratings']:
41.             if rating['method'] == 'CVSSv2':
42.                 has_cvssv2 = True
43.                 score_CVSS.append(rating['score'])
44.                 severity_CVSS.append(rating['severity'])
45.                 method.append(rating['method'])
46.                 vector.append(rating['vector'])
47.                 break
48.
49.             else:
50.                 score_CVSS.append(None)
51.                 severity_CVSS.append("NA")
52.                 method.append("NA")
53.                 vector.append("NA")
54.
55. db={'CVE': id, 'SOURCE_NAME': source_name, 'SOURCE_URL': souce_url, 'CVSS_SCORE':
score_CVSS, 'CVSS_SEVERITY': severity_CVSS, 'METHOD': method, 'VECTOR': vector, 'DESCRIPTION':
description}
56.
```

```
57. df=pd.DataFrame(db)
58.
59. #Variável  tipo de componente
60.
61. tipo = 'API'
62. tipo.strip().split(' ')
63. df.insert(1,"COMPONENT_TYPE",value=tipo)
64.
65. #Exportar output para csv
66. df.to_csv('/Users/alexandre.penela/Documents/Report Extration/Dependency
Track/Project_A/Project_A_api.csv', index=False)
```

Listagem 20 – *Script* completo de extração de dados do Dependency Track

Anexo E – Tabelas de atributos finais

Este anexo apresenta as tabelas com os atributos finais selecionados de cada uma das ferramentas.

Tabela 4 – Atributos finais do Insider

ATRIBUTOS	DESCRIÇÃO	EXEMPLO	TIPO DE DADOS
VULNERABILITY_ CLASS	Classe da vulnerabilidade	Low	Varchar
DESCRIPTION	Descrição da vulnerabilidade	File contains sensitive information written directly, such as usernames, passwords, keys, etc.	Varchar
FILE	Ficheiro do código fonte onde foi localizada a vulnerabilidade	CustomHeaders.java (14:32)	Varchar

CLASS_MESSAGE	*Path* do ficheiro onde a vulnerabilidade foi identificada	src/main/java/com/xxx/apiconfiguration/CustomHeaders.java (14:32)	Varchar
CVSS	Pontuação *Common Vulnerability Scoring System*	7.4	Decimal
CWE	*Common Weakness Enumeration*	CWE –312	Varchar
METHOD	Causa da vulnerabilidade	public static final String AUTHORIZATION = "Authorization";	Varchar
RECOMENDATION	Recomendação para resolução da vulnerabilidade	Credentials must not be stored in the Git code or repository. There are “Secrets Management” solutions that can be used to store secrets or use Pipeline resources.	Varchar

Tabela 5 – Atributos finais do OWASP Zap

ATRIBUTOS	DESCRIÇÃO	EXEMPLO	TIPO DE DADOS
VULNERABILITY_ CLASS	Classe da vulnerabilidade	Low	Varchar
ALERT	Alerta emitido	Incomplete or No Cache-control Header Set	Varchar
ALERT_REFERENCE	Número de referência do alerta	10015	Inteiro
DESCRIPTION	Descrição da vulnerabilidade encontrada	The cache-control header has not been set properly or is missing allowing the browser and proxies to cache content.	Varchar
CONFIDENCE	Nível de confiança do scanner se o alerta emitido é ou não real	Medium	Varchar

CWE_ID	Identificador *Common Weakness Enumeration* da vulnerabilidade	525	Inteiro
EVIDENCE	Evidência encontrada pelo scanner	no-cache	Varchar
MESSAGE_ID	Identificador interno da mensagem do alerta	3	Inteiro
NAME	Nome da vulnerabilidade detetada	Incomplete or No Cache-control Header Set	Varchar
PARAM	Parâmetro específico da aplicação onde foi encontrada a vulnerabilidade	Cache-Control	Varchar

PLUGIN_REF	Referência interna do *plugin* utilizado para identificar a vulnerabilidade	10015	Inteiro
REFERENCE	Referências associadas à vulnerabilidade	http://projects.webappsec.org/w/page/13246936/Information%20Leakage	Varchar
SOLUTION	Solução do *software* para resolução de vulnerabilidade	Whenever possible ensure the cache-control HTTP header is set with no-cache no-store must-revalidate.	Varchar
TAGS	*Tags* associadas ao alerta	WSTG-v42-ATHN-06: https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/04-Authentication_Testing/06-Testing_for_Browser_Cache_Weaknesses	Varchar

URL	URL onde foi encontrado a vulnerabilidade	https://test-xxxxx.xxxxxxx.corp	Varchar
WASC_REF	Referência da vulnerabilidade pela *Web Application Security Consortium*	13	Inteiro

Tabela 6 – Atributos finais do Dependency Track

ATRIBUTOS	DESCRIÇÃO	EXEMPLO	TIPO DE DADOS
COMPONENT_ TYPE	Tipo de componente analisado	API	Varchar
CVE	Código Common Vulnerability and Exposures da vulnerabilidade	CVE-2022-42003	Varchar
SOURCE_NAME	Nome da base de dados utilizada para identificar a vulnerabilidade	NVD	Varchar

SOURCE_URL	URL da base de dados	https://nvd.nist.gov/	Varchar
CVSS_SCORE	Pontuação *Common Vulnerability Scoring System*	2.1	Decimal
CVSS_SEVERITY	Classe de severidade *Common Vulnerability Scoring System* da vulnerabilidade	Medium	Varchar
METHOD	Método de utilizado na classificação da vulnerabilidade	CVSS2	Varchar

VECTOR	Vetor do método de classificação associado á vulnerabilidade	(AV:L/AC:L/Au:N/C:P/I:N/A:N)	Varchar
DESCRIPTION	Descrição da vulnerabilidade	A vulnerability was found in quarkus-core. This vulnerability occurs because the TLS protocol configured with quarkus.http.ssl.protocols is not enforced, and the client can force the selection of the weaker supported TLS protocol.	Varchar

Tabela 7 – Atributos finais das *features* do Cypress

ATRIBUTOS	DESCRIÇÃO	EXEMPLO	TIPO DE DADOS
NAME	Nome da *feature* testada	admin accept merchant invitation	Varchar
DESCRIPTION	Descrição da *feature* testada	After receiving invites, the invited person should be able to become a merchant	Varchar

STATUS	Estado da *feature*	Failed	Varchar
SCENARIO_ APPROVED	Número de cenários aprovados na *feature*	22	Inteiro
SCENARIO_ FAILED	Número de cenários reprovados na *feature*	1	Inteiro
DURATION	Duração dos testes da *feature* (em segundos)	62	Inteiro
DATE	Data do relatório em formato dd/mm/aaaa	09/10/2023	Data

Tabela 8 – Atributos finais dos cenários do Cypress

ATRIBUTOS	DESCRIÇÃO	EXEMPLO	TIPO DE DADOS
FEATURE_ID	ID da *feature* ao qual o cenário testado pertence	1	Inteiro
NAME	Nome do cenário testado	After receiving invites, the invited person should be able to become a merchant	Varchar
STATUS	Estado do cenário	Failed	Varchar
STEPS_ APPROVED	Número de *steps* aprovados no cenário	13	Inteiro
STEPS_ FAILED	Número de *steps* reprovados no cenário	11	Inteiro
DURATION	Duração dos testes do cenário (em segundos)	62	Decimal
DATE	Data do relatório em formato dd/mm/aaaa	09/10/2023	Data

Tabela 9 – Atributos finais dos *steps* do Cypress

ATRIBUTOS	DESCRIÇÃO	EXEMPLO	TIPO DE DADOS
SCENARIO_ID	ID do cenário ao qual o *step* testado pertence	1	Inteiro
KEYWORD GHERKIN	*Keyword Gherkin* do *step*	After receiving invites, the invited person should be able to become a merchant	Varchar
STEP	*Step* testado	Failed	Varchar
STATUS	Estado do *step*	13	Inteiro
DURATION	Duração dos testes do *step* (em segundos)	62	Decimal
DATE	Data do relatório em formato dd/mm/aaaa	09/10/2023	Data

Anexo F – Propriedades das Tabelas da Base de Dados

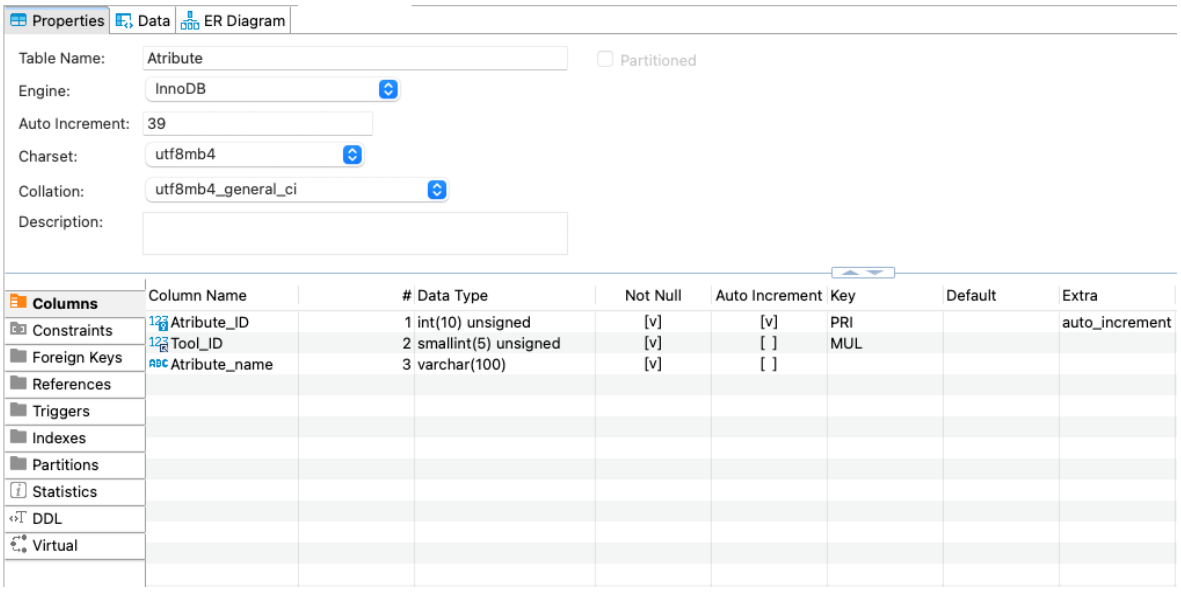


Figura 33 – Propriedades da tabela *Attribute*

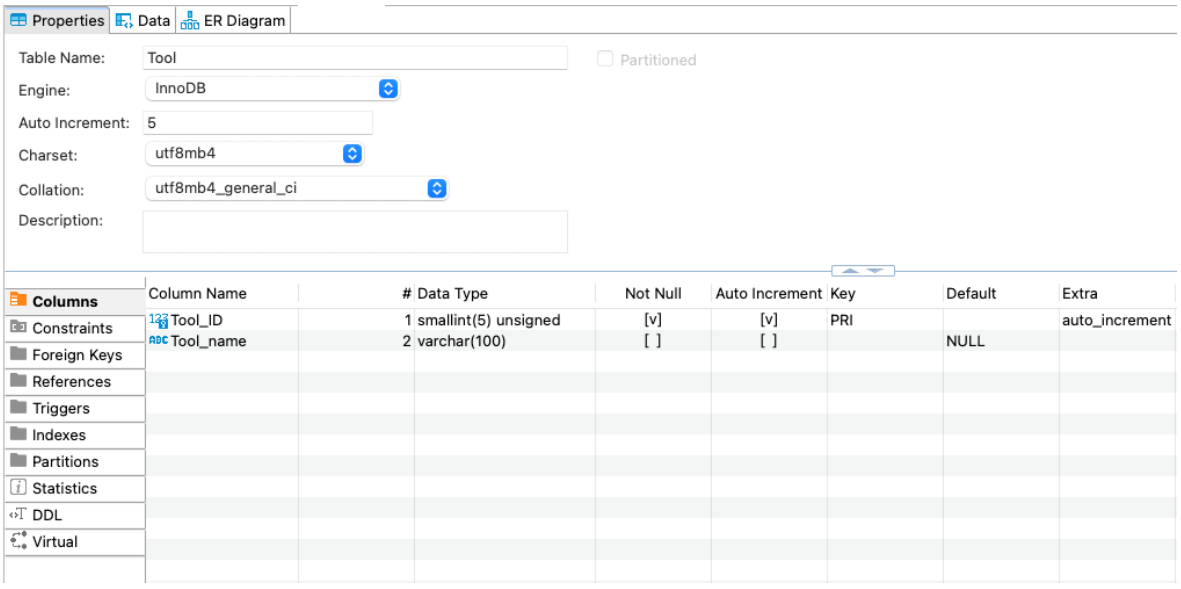


Figura 34 – Propriedades da tabela *Tool*

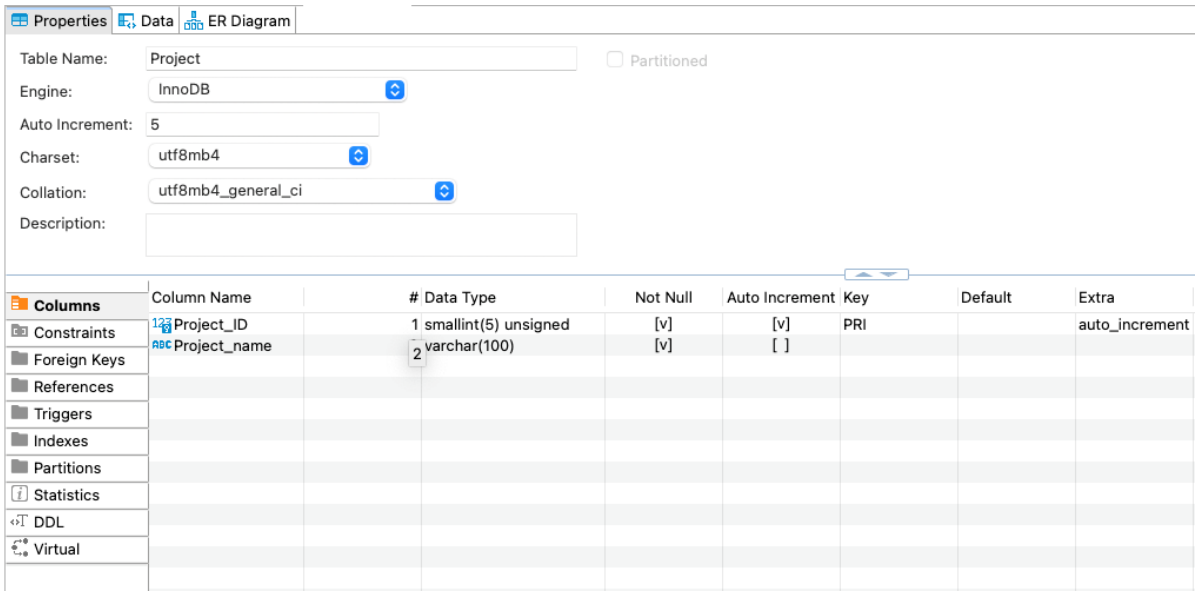


Figura 35 – Propriedades da tabela *Project*

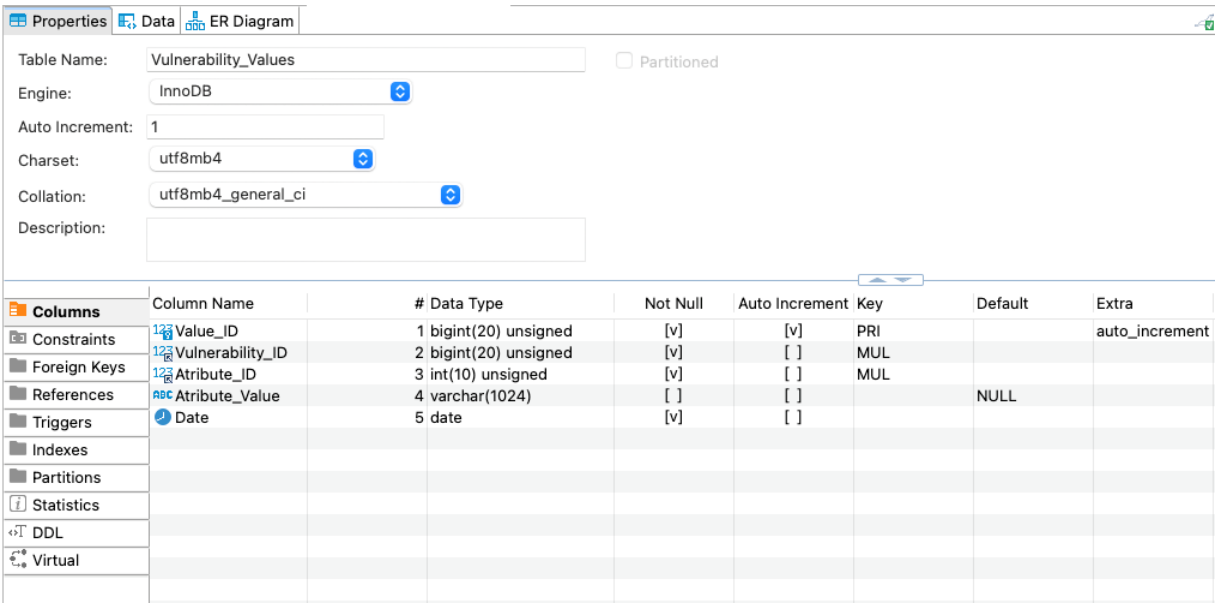
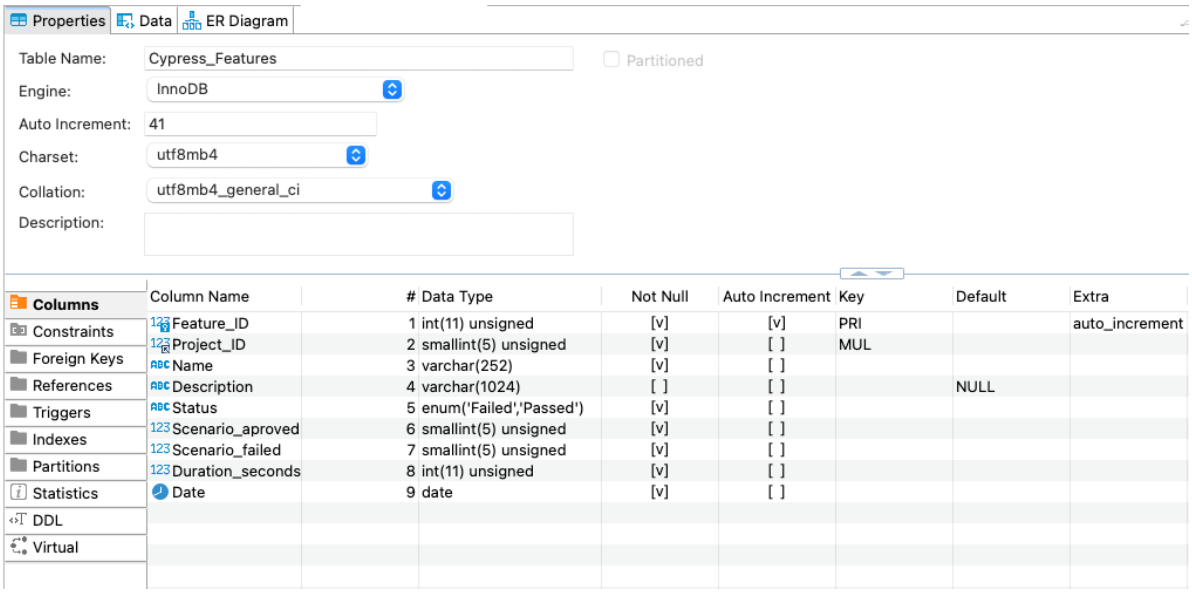
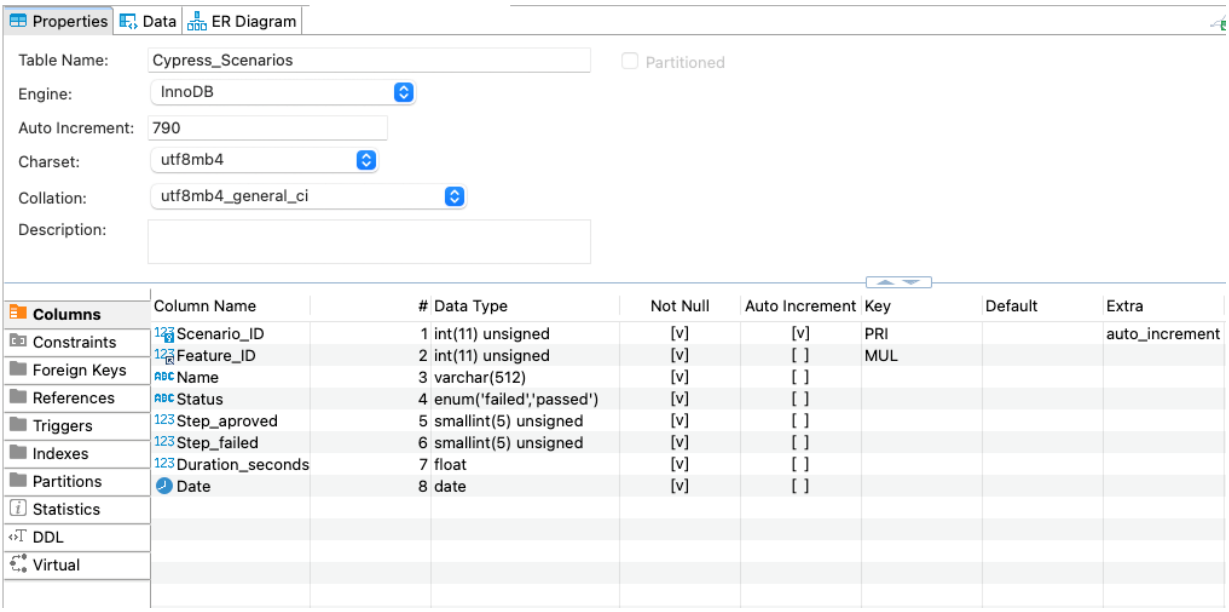


Figura 36 – Propriedades da tabela *Vulnerability_Values*



Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra
Feature_ID	1	int(11) unsigned	[v]	[v]	PRI		auto_increment
Project_ID	2	smallint(5) unsigned	[v]	[ ]	MUL		
Name	3	varchar(252)	[v]	[ ]			
Description	4	varchar(1024)	[ ]	[ ]		NULL	
Status	5	enum('Failed','Passed')	[v]	[ ]			
Scenario_aproved	6	smallint(5) unsigned	[v]	[ ]			
Scenario_failed	7	smallint(5) unsigned	[v]	[ ]			
Duration_seconds	8	int(11) unsigned	[v]	[ ]			
Date	9	date	[v]	[ ]			

Figura 37 – Propriedades da tabela *Cypress_Features*



Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra
Scenario_ID	1	int(11) unsigned	[v]	[v]	PRI		auto_increment
Feature_ID	2	int(11) unsigned	[v]	[ ]	MUL		
Name	3	varchar(512)	[v]	[ ]			
Status	4	enum('failed','passed')	[v]	[ ]			
Step_aproved	5	smallint(5) unsigned	[v]	[ ]			
Step_failed	6	smallint(5) unsigned	[v]	[ ]			
Duration_seconds	7	float	[v]	[ ]			
Date	8	date	[v]	[ ]			

Figura 38 – Propriedades da tabela *Cypress_Scenarios*

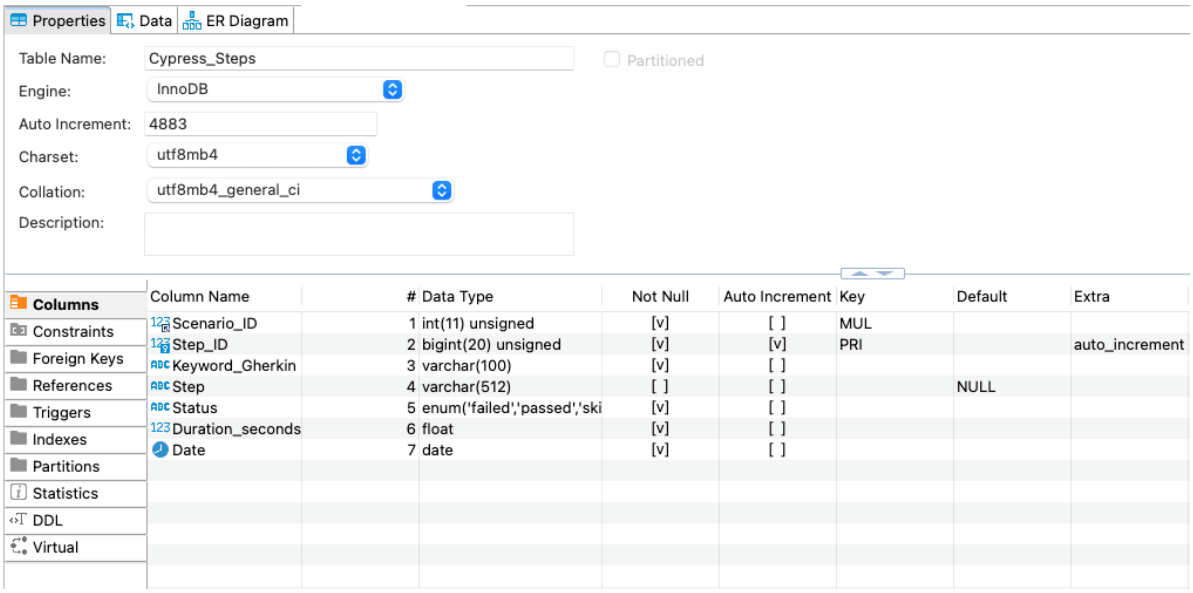


Figura 39 – Propriedades da tabela *Cypress_Steps*

Anexo G – Inquérito SUS

Este anexo apresenta um exemplar do inquérito SUS utilizado na validação da solução desenvolvida.

Teste da solução de visualização de dados no contexto da segurança de Aplicações

No âmbito do estágio curricular do mestrado em Ciência de Dados, gostaria de contar com a sua colaboração para testar e avaliar esta ferramenta de visualização de dados desenvolvida.

O objetivo deste teste é obter feedback sobre a usabilidade da solução através do System Usability Scale (SUS), uma ferramenta amplamente utilizada para avaliar a facilidade de uso de sistemas e produtos.

A sua participação consiste apenas em utilizar a solução de visualização de dados fornecida e responder a um breve questionário que contém 10 perguntas simples e diretas. As perguntas de resposta aberta no final do documento têm como intuito recolher feedback para compreender possíveis limitação que a solução apresenta e potenciais pontos de melhoria que poderão vir a ser implementadas futuramente.

O tempo estimado desta tarefa é de aproximadamente 15 minutos. A sua contribuição é essencial para a melhoria contínua da ferramenta, garantindo que ela atenda às necessidades dos utilizadores de forma eficaz e intuitiva.

O questionário é confidencial e utilizado exclusivamente para fins académicos e de melhoria da ferramenta desenvolvida.

Agradeço desde já a sua colaboração e estou à disposição para quaisquer esclarecimentos adicionais.

Alexandre Penela

Job Title: _____

	Strongly Disagree				Strongly Agree
I think that I would like to use this system frequently.	1	2	3	4	5
I found the system unnecessarily complex.	1	2	3	4	5
I thought this system was easy to use.	1	2	3	4	5
I think that I would need the support of a technical person to be able to use this system.	1	2	3	4	5
I found the various functions in this system were well integrated.	1	2	3	4	5
I thought there was too much inconsistency in this system.	1	2	3	4	5
I would imagine that most people would learn to use this system very quickly.	1	2	3	4	5
I found this system very cumbersome to use.	1	2	3	4	5
I felt very confident using this system.	1	2	3	4	5
I needed to learn a lot of things before I could get going with this system.	1	2	3	4	5